ICDAR 2019 CROHME + TFD: Competition on Recognition of Handwritten Mathematical Expressions and Typeset Formula Detection

Mahshad Mahdavi, Richard Zanibbi
Document and Pattern Recognition Lab
Rochester Institute of Technology
Rochester, NY, USA
mxm7832@rit.edu
rxzvcs@rit.edu

Harold Mouchère,
Christian Viard-Gaudin
LS2N - UMR CNRS 6004
University of Nantes
Nantes, France
harold.mouchere@ls2n.fr
christian.viard-gaudin@ls2n.fr

Utpal Garain
Computer Vision and Pattern Recognition Unit
Centre for Artif. Intel. and Mach. Leaning
Indian Statistical Institute
Kolkata, India
utpal@isical.ac.in

***Abstract*—We summarize the tasks, protocol, and outcome for the 6th Competition on Recognition of Handwritten Mathematical Expressions (CROHME), which includes a new formula detection in document images task (+ TFD). For CROHME + TFD 2019, participants chose between two tasks for recognizing handwritten formulas from 1) online stroke data, or 2) images generated from the handwritten strokes. To compare LaTeX strings and the labeled directed trees over strokes (label graphs) used in previous CROHMEs, we convert LaTeX and stroke-based label graphs to label graphs defined over symbols (symbol-level label graphs, or *symLG*). More than thirty (33) participants registered for the competition, with nineteen (19) teams submitting results. The strongest formula recognition results were produced by the USTC-iFLYTEK research team, for both stroke-based (81%) and image-based (77%) input. For the new typeset formula detection task, the Samsung R&D Institute Ukraine (Team 2) obtained a very strong F-score (93%). System performance has improved since the last CROHME - still, the competition results suggest that recognition of handwritten formulae remains a difficult structural pattern recognition task.**

***Index Terms*—mathematical expression recognition, handwriting recognition, formula detection, performance evaluation**

I. Introduction

During its history, the CROHME competition has advanced the state-of-the-art for handwritten math recognition systems, and produced a standard benchmark for online handwritten math recognition research. An IJDAR paper summarizing the outcomes and innovations in evaluating handwritten mathematical recognition during the first four years of the competition (2011-2014, 2016) is available [1], [2]. CROHME data has been used by research groups from around the world.

This sixth edition of CROHME was organized to continue encouraging activities in handwritten math recognition research, and to improve the available data, tools and benchmarks for research in this area. In CROHME 2019, there are three tasks: 1) online handwritten formula recognition (from strokes), 2) offline handwritten formula recognition (from images generated using strokes), and 3) typeset formula detection in document images.

To accommodate the growing number of encoder-decoder formula recognition systems producing LaTeX as output, in this edition we consider only symbolic formula structure, rather than in previous CROHMEs where systems were evaluated at the stroke level. For this purpose, we developed a new Symbolic Label Graph (symLG) representation that can be used with the existing evaluation tools for the competition (LgEval and CROHMELib).

Thirty-three (33) groups and individuals registered for the competition, and nineteen (19) groups submitted results. For the main task (online handwritten formula recognition), the highest recognition rate is 13% higher than in CROHME 2016 (67.65% vs. 80.73%). While encouraging, these rates suggest that recognizing handwritten math remains a difficult problem, likely due to the high number of symbol classses (101 for CROHME), and the complex two dimensional structure of math, which can include recursive and hierarchical structures.

In the following Sections we describe the competition tasks, dataset collection and encodings, evaluation metrics and tools, system descriptions, results, and then provide a brief conclusion.

II. Tasks

Task 1. Online Handwritten Formula Recognition. For the main task in CROHME, participants convert a list of handwritten strokes captured from a tablet or similar device to a Symbol Layout Tree (SLT) [3]. Participating systems are ranked based on the number of correctly recognized formulas (expression rate).

- **Task 1a (symbols)**: subtask where participants recognize isolated symbols, including 'junk' (invalid symbols). Ranked by symbol recognition rate.
- **Task 1b (parsing from provided symbols)**: subtask where participants parse formulas from provided symbols (stroke groups + labels). Ranked by expression rate.

Task 2. Offline Handwritten Formula Recognition. Strokes from the handwritten formulas in Task 1 are used

2379-2140/19/$31.00 ©2019 IEEE
DOI 10.1109/ICDAR.2019.00247

TABLE I: CROHME 2019 DATA SETS

Tasks	Training	Validation	Test
Formulae (1, 2)	Train 2014 + Test 2013 + Test 2012 9993 expr	Test 2014 986 expr	Test 2019 1199 expr
Symbols (1a, 2a)	Train 2014 + Test 2013 + Test 2012 180440 symbols + junks	Test 2014 18435 symbols + junks	Test 2016 15483 symbols + junks
Structure (1b, 2b)	Train 2014 + Test 2013 + Test 2012 9993 expr	Test 2014 986 expr	Test 2016 1147 expr
Formula Detection (3)	36 rendered PDFs (600dpi): 569 pages 26395 formula regions Character BBs and labels	n/a	10 rendered PDFs (600dpi): 236 pages 11885 formula regions Character BBs *without* labels

to render images. Participants then convert these handwritten formula images to a Symbol Layout Tree. For evaluation, the same evaluation tools are used as for Task 1. Again, participants are ranked by the expression rate of their system.

- **Task 2a (symbols)**: subtask where participants recognize isolated symbols, including 'junk' (invalid symbols). Ranked by symbol recognition rate.
- **Task 2b (parsing from provided symbols)**: subtask where participants parse formulas from provided symbols (bounding boxes + labels). Ranked by expression rate.

Task 3. Detection of Formulas in Document Pages. Given a document page along with the bounding boxes of characters on that page (as are available for born-digital PDF files), participating systems identify formulas using bounding boxes. Evaluation is performed using Intersection-over-Union (IoU, or equivalently the Jaccard similarity coefficient), and systems are ranked based on their F-measure after matching output formula boxes to ground truth formula regions.

III. DATASETS AND FORMULA ENCODINGS

In this Section we describe data used in the competition, how it was collected, and the encodings used. Table I summarizes the datasets used for the competition.

Handwritten Formulas: Input Data. For Task 1, we use online data in the same InkML and Label Graph (LG) file formats from previous CROHMEs. Strokes are defined by lists of (x,y) coordinates, representing sampled points as a stroke is written. Groupings of strokes into symbols, symbol labels, and formula structure are provided in both the InkML and LG formats. In InkML structure is represented using Presentation MathML (an XML-based representation), while in LG a simpler CSV-based representation is used. In both cases, formula structure is represented by a Symbol Layout Tree, as seen in Figure 1(b). Roughly speaking, this format represents the appearance of a formula by the placement of symbols on the different writing lines of the expression. Spatial relationships between symbols (e.g, 'R' for adjacent-at-right) are indicated using edge labels.

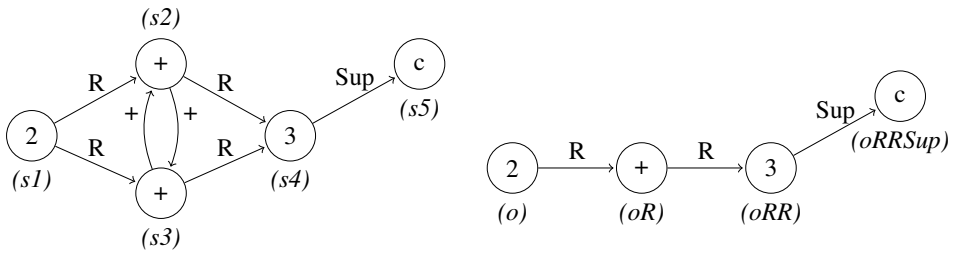
For Task 2, the offline formula data is provided as greyscale images. These were rendered automatically from the online data with 1000×1000 pixels with 5 pixels of padding. This format is used for the main task (Task 2) and Task 2b. For the isolated symbol sub-task, isolated symbols are rendered at 28×28 pixels with the same amount of padding (5 pixels). The resolution of the inputs files is fixed for Task 2, but participants were welcome to resize the original images using pre-processing methods of their choice.

Symbol Layout Graph (symLG) Formula Representation. The stroke-based LG files used in previous CROHMEs allow all segmentation, classification, and structural errors to be identified unambiguously, even when segmentations disagree [1]. However, with the success of encoder-decoder-based systems that generate LaTeX output, a new representation is needed - these systems do not output information about stroke segmentation or the location of symbols in the input, instead producing Symbol Layout Trees directly.

Figure 1 shows two graph representations for the same expression '$2+3^c$.' In the Stroke Label Graph (LG), there are 5 nodes (one per stroke), and edges represent segmentation and spatial relationships between pairs of strokes (including 'no relationship'). For the Stroke Label Graph, node identifiers are for individual strokes. In the Symbol Label Graph (symLG), there are 4 nodes (one per symbol) and edges represent relationships between symbols (no segmentation information is provided). For symLG, node identifiers are constructed from the sequence of relation labels on the path from root to the symbol. For example, 'c' in Figure 1 has the identifier 'oRRSup' (origin/root, Right, Right, Superscript).

To compute the similarity of two Symbol Layout Trees in our symLG representation, we use an adjacency matrix. Labels on the diagonal define symbol labels, while off-diagonal elements represent spatial relationships between parent and child symbols. Using this representation, we can determine how formulas in an SLT representation differ in structure and symbol labels, but not the correspondence between symbols and relationships in a symLG file and the input data (i.e., strokes/images). Still, the symLG representation allows existing metrics and tools designed for evaluation of stroke-level LG files at the symbol level to be used directly. We note that symLG is closely related to the tree-based symbolic representation from earlier CROHME competitions [1], but permits more detailed error analysis.

Formula Data Collection. We used the labeled handwritten formulae from previous CROHMEs that are publicly available



(a) Stroke Label Graph (LG) **(b) Symbol Label Graph (symLG)**

Fig. 1: Different graph representations for formula '$2 + 3^c$' written using 5 strokes. Node identifiers are shown in brackets.

[2] to generate training and validation sets for the formula recognition tasks. The new formula test set was produced by the organizers at the University of Nantes and the Indian Statistical Institute (ISI). Three input devices were used, with different sampling and scaling: TabletPC with pen-based touch screen, a whiteboard, and a graphic tablet. We used arXiv papers from 2002 and 2003 made available through the KDD 2003 Cup [4]. A corpus of 1207 expressions was selected, according to the same expression grammar and symbol frequency constraints used to create the CROHME 2016 Test set using the algorithm described in [1]. 1200 expressions were produced by 80 writers (15 expressions each), with only one expression being discarded. The data was annotated using a semi-automatic process [2]. Some expressions were cleaned, removing noisy strokes or splitting connected symbols, and sometimes the ground-truth LaTeX string was updated to fit what was actually written.

We expanded the CROHME 2019 Training set by adding previous test sets (from 2012 and 2013, see Table I). The Validation set for 2019 is the CROHME 2014 Test set. Test sets for Tasks 1a/b and Tasks 2a/b were the CROHME 2016 Test set.

Formula Detection Data. For typeset formula detection, we modified the GTDB-dataset[1] created by the *infty* group (Masakazu Suzuki et al. [5]). Documents in the collection are PDFs taken from scientific journals and textbooks with different font faces and notation styles. GTDB provides ground-truth at the character and formula level (including SLTs) in CSV format, with carefully annotated displayed (offset) formulas and formulas within text lines (inline formulas). Nearly 25% of the annotated formulas are single symbols (e.g., λ). The organizers at RIT modified the ground-truth data to compensate for differences in scale and translation found in the publicly available versions of PDFs in the collection. Details may be found in Table I. A script was provided to compile and render the competition PDFs at 600 dpi. In addition to the document images, character bounding boxes were provided; in the training set, character labels, and indication of whether a character belongs to a formula region were also provided.

[1]https://github.com/uchidalab/GTDB-Dataset

IV. Evaluation Metrics and Tools

Submission. Beginning with CROHME 2016, we have used an online submission and evaluation tool[2] implemented in Django. Multiple submissions were permitted for each team, with the highest recognition rates used to rank systems. After each submission, the performance metrics were computed, and a leaderboard visible to all participants was updated in real time. For the Test sets, only recognition rates/F-scores were visible to participants.

Tools. Updated versions of the CROHMELib and LgEval libraries used for the competition were made available to participants offline. These tools support converting between LG and InkML, and computing performance metrics [1]. For formula detection, we developed a tool that ranked based on Intersection-over-Union (IoU). We also provided tools to let participants convert their outputs from MathML, LaTeX,or LG into SymLG.

Formula Recognition Metrics. LgEval metrics include formula and symbol recognition rates, along with recall and precision metrics for detection (segmentation) and detection + classification for both symbols and relationships [1]. Recognition systems that produce LaTeX strings or stroke-level Label Graphs (LGs) both have their outputs converted to symLG (see Figure 1). This allows systems producing stroke-level and symbol-level results to be compared directly, albeit with a loss in the stroke segmentation information provided in the stroke-level representation. The symLG representation allows us to identify errors as symbol classification errors, relationship classification errors, or structure errors.

Note that this change in representation has several impacts on the results; in particular, segmentation information cannot be calculated - it is possible for a formula to be recognized with the correct Symbol Layout Tree, but without correctly segmenting symbols. This makes the expression rate less strict than at the stroke-level. Also, because labels on relationship paths identify symbols, when symbols do not appear at expected locations in the output they are treated as missing ('ABSENT' in LgEval), which leads to an underestimate of symbol recall.

Recognition rates were used to rank systems for Tasks 1, 2, and their subtasks.

[2]http://crohme2019.cs.rit.edu

Formula Detection Metric. We use an F-score based on matches with an $IoU \geq 0.75$ to rank systems. For each ground truth box in a document page, if there is a detection box that has an IoU above the given threshold, the GT box is considered to be detected correctly. We enforce a one-to-one mapping, where each output bounding box is assigned to the GT bounding box with the highest IoU score, with each GT box matched to at most one output bounding box. We used an implementation by Padilla[3] to calculate intersections for each pair of boxes. We then calculate precision, recall and F-scores. In the offline tool, all boxes missed in the output are automatically listed with the corresponding IoU score for error analysis. For each system, we calculate both a coarse detection score ($IoU \geq 0.5$) and a fine detection score ($IoU \geq 0.75$).

V. Participating Methods

The system descriptions in this Section are provided by the CROHME 2019 participants (note: some are missing). **Systems are organized according by their order of appearance in Tables II and IV.** Nineteen teams submitted results for CROHME 2019, from which 9 teams participated in Task 1 and its sub tasks, 8 teams participants in Task 2 and its sub tasks, and finally 4 teams participated in Task 3. *Tasks participated in are indicated in square brackets before each team name.*

[1, 2] USTC-NELSLIP and iFLYTEK Research. We utilize attention based encoder-decoder models: an RNN-based encoder for online formula recognition [6], and a CNN-based encoder for offline formula recognition [7]. We strengthen the online encoder by using a CNN-RNN architecture, and strengthen the offline encoder using an enhanced DenseNet architecture. For Task 1 we combine the online and offline recognition models. For Task 2, we preprocess the images by first extracting the image foreground, resize it using the estimated average height of each symbol based on information in original images. We train using only the official training dataset, and employ a data augmentation method [8] to alleviate the problem of limited training data. Finally, we use the text dataset provided in the NTCIR-12 MathIR dataset [9] to train a single layer RNN-based language model, and combine it with our encoder-decoder models [6].

[1, 1b] Samsung R&D Institute Ukraine - Team 1. This system is a deep-learning solution that employs multi-task recurrent neural networks (RNNs) for character recognition in combination with preprocessing and re-ordering, and a Probablistic Context-Free Grammar (PCFG) in combination with spatial relation analysis and language models. For Task 1b, we used a PCFG in combination with spatial relation analysis and language models. In order to cover the complex 2D structure of Handwritten Math equations, we have introduced two different bigram language models: language sequence model and language relation model. For parsing, we introduced heuristics into bottom-up parsing and the dynamic programming-based CYK parsing algorithm. The first heuristic exploits the concepts of dominance and mutual visibility for character candidates. The second heuristic applies dynamic pruning to the search area depending upon detected character candidates. We used an additional dataset to train our system for both tasks [10].

[1] MyScript. The MyScript Math recognizer system is built on the principle that segmentation, recognition, and interpretation have to be handled concurrently and at the same level in order to produce the best candidates. The recognition engine analyzes the spatial relationships between all parts of the equation, in conformity with rules laid down in its grammar, to determine the segmentation of all of its parts. The grammar is defined by a set of rules describing how to parse an equation, each rule being associated with a specific spatial relationship. The symbol recognizer combines Neural Nets evaluating hypotheses with static and dynamic characteristics, and a BLSTM trained with the CTC algorithm, which processes the ink signal directly. The recognition also combines statistical and LSTM-based language models, to evaluate the contextual probabilities of symbols in the equation. This recognizer was trained specifically on the data provided for the online formulas recognition task (CROHME Train 2014, Test 2012, Test 2013). We also built a second system, by combining this specific recognizer with our public Math recognition engine, trained on more generic data. The combination improves the accuracy by solving some ambiguous cases.

[1, 2] Sun Yat-Sen University. This system recognizes offline formulas by extracting strokes from the input image and then performing online recognition. First, input images are re-scaled to 1500x1500 and binarized using Sauvola's method. Second, a thinned skeletonization of the image is decomposed into junctions and segments, with segments that are too short being discarded. Third, segments pairs sharing a junction with a minimal difference in direction are linked repeatedly, after which double-traced strokes are fixed using heuristic rules. Fourth, strokes are sorted using a recursive X-Y cut and topological sort. Finally, after extracted strokes are recognized by MyScript Interactive Ink (version 1.3) using a customized grammar and DPI of 576, exported MathMLs are converted to CROHME's conventions and then converted to symLGs [11].

[1, 3] Samsung R&D Institute Ukraine - Team 2. Our formula detector is based on graph-theoretic methods for determining the position of multi-character formulae, in combination with statistical and context-recognition-based approaches for detecting single-character mathematical symbols inside text [12].

[1, 2] CASIA/NLPR PAL Group We entered two systems: PAL, and PAL-v2 which extends our previous work [13]. The attention-based encoder-decoder model in PAL-v2 is trained using official data only. Training data is rendered as 1000x1000 offline images using the official tool, after which we extract the image foreground and normalize the height to 64 pixels, using dilation/erosion to thicken the strokes. We augment the training data set using rotations, perspective shift, distortion, and bevel, as well as the decomposition operation

[3]https://github.com/rafaelpadilla/Object-Detection-Metrics

introduced by Le et al. [8]. This expanded the training data to 330k images, which are then used for Paired Adversarial Learning [13]. An ensemble of 6 models with different initializations produced the PAL-V2 results. We did not use an additional language model to assist with recognition. The PAL architecture for Task 2 is the same as PAL-v2, but is trained on both the augmented training set and extra data *without* Paired Adversarial Learning. The extra data was obtained using an additional 70k formulas from the Wikipedia formula corpus, from which we generated 120k formula images. These 120k Wiki formulas were combined with the original training data, and then augmented to produce 3 times more data using the procedure described above.

[1] MathType. The system is composed of a convolutional encoder and an attention-based LSTM decoder. The online input coordinates are encoded as an image, where additional channels account for trajectory information. It is a model designed to be fast and use low memory for deployment in real-time scenarios. We trained the system using the competition dataset and also with additional private data.

[1, 2] Tokyo University of Agriculture and Technology. Task 1: The system includes three main tasks: symbol segmentation, symbol recognition, and structure recognition. The symbol recognizer is a combination of LSTM models extracting features for both online and offline patterns (converted from online patterns). We improved the structure recognition system and added grammar rules from the version that participated in CROHME 2016. Additional detail of our system are presented elsewhere [14]. Task2: For offline handwritten mathematical expression recognition, we employ an attention-based residual sequence-to-sequence (ARseq2seq) model [15]. Our ARseq2seq model consists of 3 parts: 1) feature extraction by CNN, 2) encoding by BLSTM, and 3) decoding by LSTM with an attention mechanism. We also introduce using the residual connection in the decoder. Details of the Arseq2seq model are presented in Ly et al. [15]. The input images are cropped and resized into a 192×192 image.

[2, 2b] University of Linz. This system is composed of two subsystems: the first is the symbol detector, and the second is the symbol parser. The symbol detector is a Faster RCNN object detection model using an Inception Resnet (v2) convolutional neural network as the backbone. The symbol parser is also implemented using a deep neural network, based on the Transformer model developed by Vaswani et al.,where instead of positional embeddings, we use bounding box embeddings.

[3] Rochester Institute of Technology - Team 1. We treat formula detection as an object recognition problem. A modified YOLOv3 model [16] containing a 53-layer CNN (Darknet-53) is used to define a 416×416 detection window, which we then pass over the large document image using a 10% stride at train and test time. A pixel-level confidence map for the document is obtained by averaging detection box confidences across window locations (i.e., average pooling). Detected formula bounding boxes are extracted from the document confidence map by thresholding followed by connected component extraction. A post-processing step is used to refine detections: math regions are clipped/grown to fit all connected-components lying within a detected formula region. The model performs well for displayed formulas in particular, with some under-segmentation for formulas spanning multiple text-lines (e.g., in derivations).

TABLE II: FORMULA RECOGNITION RESULTS

	Structure + Symbol Labels			Structure
2019 Test Data	Correct	$\leq$ 1 s.err	$\leq$ 2 s.err	**Correct**
Task 1: Strokes				
USTC-iFLYTEK	**80.73**	**88.99**	**90.74**	**91.49**
Samsung R&D 1	79.82	87.82	89.15	89.32
MyScript	79.15	86.82	89.82	90.66
Sun Yat-Sen U.	77.40	85.82	87.99	88.82
Samsung R&D 2	65.97	77.81	81.73	82.82
PAL-v2	62.55	74.98	78.40	79.15
MathType	60.13	74.40	78.57	79.15
TUAT	39.95	52.21	56.54	58.22
Task 2: Images				
USTC-iFLYTEK	**77.15**	**86.82**	**88.99**	**89.49**
PAL	71.23	80.31	82.65	83.82
Sun Yat-Sen U.	65.22	78.48	83.07	84.90
PAL-v2	62.89	74.98	78.40	79.32
SCST-USTC	62.14	75.06	78.23	78.32
Univ. Linz	41.49	54.13	58.88	60.02
TUAT	24.10	35.53	43.12	43.70
2016 Test Data				
Task 1B: Strokes with Provided Symbols				
Samsung R&D 1	**92.94**	**93.11**	**93.20**	**93.20**
fvg	85.88	85.88	85.96	85.88
Task 2B: Images with Provided Symbols				
Univ. Linz	81.34	81.87	82.30	82.21

s.err: symbol class label errors

[3] Rochester Institute of Technology - Team 2. We detect formulas using a single shot multi-box detector (SSD512) with two major modifications. First, we use wider default boxes in addition to default boxes from the original SSD architecture. Wider default boxes help us detect wide formulas. Second, we use focal loss instead of cross-entropy loss for training. Adding wide default boxes increased the number of default boxes to over 45k, which in turn increases the imbalance between positive and negative examples. Therefore, we use the focal loss to give increased weight to ambiguous examples near decision boundaries. We use a sliding window approach, splitting the original 600dpi image into images of size 512×512 using a 10% stride. Finally, detections obtained from SSD are used in a voting-based pooling strategy to get math bounding boxes at the scale of the original image.

VI. RESULTS

In this Section we summarize the results of the competition. No teams participated in the symbol recognition tasks (Tasks 1a and 2a), so we do not discuss them further.

Formula Recognition (Tasks 1 and 2). Table II shows the expression recognition rate for each system, with systems sorted in decreasing order of their correct expression rate. Also shown are the number of expressions correctly recognized when 1 to 2 errors in symbol class labels are permitted, along with the number of formulas with the correct

TABLE III: FORMULA REC. ERRORS (2019 TEST DATA)

	Symbols			Symbol Pairs (w. Relation)		
	1st	2nd	3rd	1st	2nd	3rd
Task 1: Strokes	**E #**	**E #**	**E #**	**E #**	**E #**	**E #**
USTC-iFLYTEK	1 70	+ 54	53	$\frac{1}{}$ 16	$\frac{}{2}$ 15	+− 13
Samsung R&D 1	70	− 69	2 62	+1 15	14	−1 10
MyScriptA	56	+ 51	48	14	−1 8	+2 8
Sun Yat-Sen U.	71	2 67	− 65	16	14	(x 10
Samsung R&D 2	120	− 98	+ 89	−1 20	+1 18	$\frac{1}{}$ 18
Task 2: Images						
USTC-iFLYTEK	1 66	+ 57	2 57	$\frac{1}{}$ 14	$\frac{}{2}$ 13	$C =$ 9
PAL	127	86	80	$\rightarrow \infty$ 21	+1 19	$= 1$ 19
Sun Yat-Sen U.	107	105	102	+1 25	−1 25	$x+$ 20
PAL-v2	−819	2 724	1 706	$\frac{1}{}$ 164	$\frac{}{2}$ 130	(x 110
SCST-USTC	944	828	818	205	153	−1 120

E: symbol/subexpresion error
Note: repeated symbols and subexpressions omitted for readability.

TABLE IV: TYPESET FORMULA DETECTION RESULTS

	IoU ≥ 0.75			IoU ≥ 0.5		
	F1	**Recall**	**Prec.**	**F1**	**Recall**	**Prec.**
Samsung R&D-2	**93.45**	**92.73**	**94.17**	93.63	92.91	94.36
RIT 2	68.29	62.49	75.29	75.41	67.00	83.14
RIT 1	60.58	58.17	63.20	71.32	68.47	74.40
michiking	16.14	13.99	19.10	31.18	27.00	36.87

recognized structure, ignoring symbol labels (i.e., treating all symbols as correct). Allowing just one labeling error, all rates increase substantially. **USTC-iFLYTEK** obtained the highest recognition rate for both strokes in task1 and images in task2. We should mention that MyScript (accidentally) withdrew a submission that obtained an expression rate of 80.82%, but this was determined after the competition was over. To be fair to other participants, we have removed this from the official results table.

A surprisingly small drop-off in recognition accuracy occurs when the stroke data is removed and raw pixels are provided in Task 2, decreasing the recognition rate from 80.73% for online recognition to 77.15% for offline recognition. A summary of confusion histograms that tabulate errors in symbol and relationship recognition is presented in Table III (for space, the largest subgraphs shown are pairs of symbols). The detailed results show that the most common error is missing symbols, which is due to symbols being identified by their absolute path: errors in structure lead to errors in symbol location and classification. Looking at the detailed confusion histogram results produced using the *confHist* tool from LgEval, in most cases, there is a shift in the structure causing missing nodes.

Typeset Formula Detection. Table IV presents the Recall and Precision rates, ranking participants by F-Score. There is a real gap between the winning system **Samsung R&D-2** and the other three participants. One factor may be that the winning system made effective use of the provided character locations which the RIT teams ignored, for example. Using a more permissive metric (IoU ≥ 0.5) substantially increases the F-score for all but the winning system.

VII. CONCLUSION

For the formula recognition tasks, end-to-end neural networks produced very strong results, winning both tasks. However, hybrid systems combining grammatical and statistical approaches still obtain results close to the state-of-the-art (e.g., MyScript). Concerning the formulae detection in printed documents, the graph-based approaches outperformed participants' deep-learning approaches. The results of the competition lead us to conclude that math recognition remains a challenging task for state-of-the-art systems.

Acknowledgments. Our sincerest thanks to all of the participants. The authors would also like to thank Puneeth Kukkadpu and Parag Mali for preparing data for the formula detection task, Soumen Kumar Koley for helping update the online submission system, and Rafael Padilla for the initial implementation of object detection metrics. This work was supported in part by the Alfred P. Sloan Foundation and the Centre for Artificial Intelligence and Machine Learning (CAIML) of ISI, Kolkata.

REFERENCES

[1] H. Mouchère, R. Zanibbi, U. Garain, and C. Viard-Gaudin, "Advancing the state of the art for handwritten math recognition: the CROHME competitions, 2011–2014," *IJDAR*, vol. 19, no. 2, pp. 173–189, 2016.

[2] H. Mouchère, C. Viard-Gaudin, R. Zanibbi, and U. Garain, "Icfhr2016 competition on recognition of handwritten mathematical expressions (crohme2016)," in *ICFHR*, Shenzhen, Chine, 2016.

[3] R. Zanibbi and D. Blostein, "Recognition and retrieval of mathematical expressions," *IJDAR*, vol. 15, no. 4, pp. 331–357, 2012.

[4] J. Gehrke, P. Ginsparg, and J. Kleinberg, "Overview of the 2003 KDD cup," *ACM SIGKDD Explorations Newsletter*, vol. 5, no. 2, pp. 149–151, 2003.

[5] 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.

[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, 2019.

[7] 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] A. D. Le, B. Indurkhya, and M. Nakagawa, "Pattern generation strategies for improving recognition of handwritten mathematical expressions," *arXiv preprint arXiv:1901.06763*, 2019.

[9] R. Zanibbi, A. Aizawa, M. Kohlhase, I. Ounis, G. Topić, and K. Davila, "NTCIR-12 MathIR task overview," in *Proc. NTCIR-12: Evaluation of Information Access Technologies*, Tokyo, 2016, 10 pp.

[10] Z. V. R. O. Zhelezniakov, D., "Acceleration of online recognition of 2d sequences using deep bidirectional lstm and dynamic programming," *15th International Work-Conference on Artificial Neural Networks (IWANN), LNCS 11507*, 2019.

[11] C. Chan, "Stroke extraction for offline handwritten mathematical expression recognition," *arXiv preprint arXiv:1905.06749*, 2019.

[12] I. Degtyarenko, O. Radyvonenko, K. Bokhan, and V. Khomenko, "Text/shape classifier for mobile applications with handwriting input," *IJDAR*, vol. 19, no. 4, pp. 369–379, 2016.

[13] J.-W. Wu, F. Yin, Y.-M. Zhang, X.-Y. Zhang, and C.-L. Liu, "Image-to-markup generation via paired adversarial learning," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2018, pp. 18–34.

[14] A. D. Le and M. Nakagawa, "A system for recognizing online handwritten mathematical expressions by using improved structural analysis," *IJDAR*, vol. 19, no. 4, pp. 305–319, 2016.

[15] C. T. N. N. T. Ly and M. Nakagawa, "An attention-based end-to-end model for multiple text lines recognition in japanese historical documents," *ICDAR*, 2019.

[16] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.