

# Clustering and user interface to assist in marking handwritten mathematical expressions

手書き数式解答の採点を支援するためのクラスタリング  
とユーザーインターフェイス

by

**VU TRAN MINH KHUONG**

18834307

A thesis submitted in conformity with the requirements for the degree of  
*Doctor of Philosophy*  
in  
*Electronic and Information Engineering*

Graduate School of Engineering  
Tokyo University of Agriculture and Technology

Under the supervision of Prof. Masaki Nakagawa and Prof. Keiichi Kaneko  
2020



## **Acknowledgments**

First, I would like to express my deepest gratitude to my supervisors, professor Masaki Nakagawa and professor Kaneko, for their guidance, motivation, and continuous support during my study. Thank you for being great mentors and diligent readers for all my papers. Your insightful and constructive comments, as well as encouragement and inspiration, have made my doctoral studies of my a wonderful and fruitful journey.

I hereby express my gratitude to the Hasegawa International Scholarship Foundation for offering me a scholarship since 2019 that helped me a lot.

I would like to thank all members and staff of Nakagawa Laboratory Kaneko Laboratory, and iLabo Company for their valuable comments and supports. Especially, I would like to acknowledge my colleagues: Dr. Nguyen Tuan Cuong, Dr. Le Duc Anh, Dr. Phan Minh Khanh, Dr. Nguyen Tuan Hung, Mr. Ung Quang Huy, and Mr. Truong Thanh Nghia for supporting me during my doctoral years.

My special thanks go to my former professor Pham The Bao for mentoring and giving the first bricks of my research career during the time at the Ho Chi Minh City University of Science, Vietnam. I also acknowledge professor Indurkha for screening my papers and giving valuable comments.

I owe big thanks to my family for raising me with constant encouragement and understanding during my recent hard time. Last but not least, I also thank all my friends for everything.

## ABSTRACT

In this thesis, the author presents some approaches to help teachers reduce the workload in marking handwritten mathematical answers in examinations and assignments. First, he introduced a semi-automatic approach of marking offline handwritten mathematical answers by clustering handwritten math expressions (HMEs). Secondly, he presented a method for generating handwritten mathematical answers (HMAs) artificially which is used to generate a synthetic dataset and made it publicly available for evaluating methods of automatic or semi-automatic marking of HMEs. Finally, the author presented an interactive user interface that can be applied in marking mathematical examinations and assignments using pen-based devices.

Specifically, chapters 1 and 2 presented the background and related works on educational assessment. The author discussed the pros and cons of approaches that aim to help scale up the marking process. Then he presented the current problems of applying offline HME recognition to mark HMAs in practice and subsequently introduced the method of clustering HMEs to overcome those problems.

In chapter 3, the author presented more details about the offline HME clustering approach which aims to reduce the marking workload of HMAs. The approach groups similar answers into clusters. Hence, it allows teachers to reduce the marking workload efficiently by marking multiple answers in a cluster with only one action. Many features from multiple levels of recognition have been investigated and evaluated. Then a method of combining those features is applied. Besides, the author proposed a measurement to estimate the marking effort concerning the number of clusters and purity of each cluster. The experimental results show that the method can reduce the workload by up to 40%.

In chapter 4, the author presented a method for generating handwritten mathematical expressions artificially which is used for generating a synthetic dataset. The method generates synthetic HMEs from LaTeX strings and utilizes handwritten patterns from an international competition's dataset which is published for the HME recognition problem. Then the author evaluated his methods on the generated dataset and made it publicly available for encouraging research in the HME clustering topic.

In chapter 5, the author designed an interactive user interface that allows users to input math expressions to a pen-based device easily. The interface shows the recognition errors from multiple

recognition levels. Hence, users can verify and correct misrecognitions by interacting with controllers of the interface. Multiple editing gestures are proposed to correct recognition errors according to multiple recognition levels. This interface can be applied for an examination that is conducted on electronic devices because of its flexibility and effectiveness.

Chapter 6 and chapter 7 presented the remaining work and concluded the research. In summary, the thesis provides some approaches to improve the assessment of handwritten answers in mathematical examinations. Though there are still some steps that need to be done before applicable in practice, these approaches show promising solutions to scale up examinations with huge size of students.

# Table of Contents

<b>1</b>	<b>Introduction.....</b>	<b>9</b>
1.1	Background.....	9
1.2	Contributions .....	9
1.3	Structures of the thesis.....	10
<b>2</b>	<b>Surveys.....</b>	<b>12</b>
2.1	Offline HME Clustering .....	12
2.1.1	<i>HME Recognition (HMER)</i> .....	12
2.1.2	<i>Online and Offline HMER</i> .....	14
2.1.3	<i>Computer-Assisted Marking Systems</i> .....	15
2.1.4	<i>Survey on Clustering Algorithms</i> .....	17
2.2	Synthetic Data Generation.....	20
2.2.1	<i>Datasets</i> .....	20
2.2.2	<i>Measurements for HMER</i> .....	20
2.3	User Interface for Inputting MEs.....	22
<b>3</b>	<b>Offline HME Clustering .....</b>	<b>23</b>
3.1	Introduction .....	23
3.2	Our Proposed Method.....	23
3.3	Multi-Level Features From an HME .....	24
3.3.1	<i>Directional features</i> .....	25
3.3.2	<i>Bag-of-symbols</i> .....	28
3.3.3	<i>Bag-of-relations</i> .....	29
3.3.4	<i>Bag-of-positions</i> .....	31
3.3.5	<i>Feature combination methods</i> .....	32
3.3.6	<i>Positional bag-of-features</i> .....	33
3.4	CNN-based Spatial Classification Features.....	34
3.4.1	<i>Multi-scale object localization and classification features</i> .....	35
3.4.2	<i>Method</i> .....	37
3.5	Marking Cost Function.....	42
3.6	Incremental refining clustering.....	45
3.7	Datasets.....	47
3.8	Experiments and Results .....	49
3.8.1	<i>Experiments on individual feature types</i> .....	49
3.8.2	<i>Experiments on feature combination</i> .....	50
3.8.3	<i>Experiments on marking cost for Dset_Mix</i> .....	52
3.8.4	<i>Experiment on CNN based Spatial Classification Features</i> .....	54
<b>4</b>	<b>Synthetic Data Generation.....</b>	<b>60</b>
4.1	Introduction .....	60
4.2	Synthesizer.....	61
4.2.1	<i>Input</i> .....	61
4.2.2	<i>Symbol Relation Tree</i> .....	62
4.2.3	<i>The layout of the Synthetic HME</i> .....	62
4.2.4	<i>Selecting and Positioning Handwritten Symbols</i> .....	64
4.2.5	<i>Root Symbol with inside relation</i> .....	65
4.2.6	<i>Output</i> .....	66
4.3	Experiments.....	66
4.4	Study of the clarity of the synthetic samples .....	67

4.5	Study of the naturalness of the synthetic samples .....	68
4.6	Study of the unnatural points in the synthetic samples.....	68
<b>5</b>	<b>Interactive User Interface for Inputting MEs .....</b>	<b>70</b>
5.1	Introduction .....	70
5.2	User interface design for displaying recognition result and misrecognition correction .....	71
5.3	Editing Gestures .....	72
5.3.1	<i>Gestures for Editing Symbol Segmentation Errors .....</i>	<i>72</i>
5.3.2	<i>Gestures for Editing Symbol Recognition Errors.....</i>	<i>73</i>
5.3.3	<i>Gestures for Editing Structure Analysis Errors .....</i>	<i>74</i>
5.4	Experiments.....	76
5.4.1	<i>Comparing two interface versions and editing gestures .....</i>	<i>77</i>
5.4.2	<i>Evaluating user experience .....</i>	<i>79</i>
<b>6</b>	<b>Future works .....</b>	<b>81</b>
<b>7</b>	<b>Conclusions.....</b>	<b>82</b>

## Table of Figures

Figure 1. The flow of the general approach for recognizing HMEs.....	12
Figure 2. Examples of error types in the symbol segmentation task .....	13
Figure 3. The flow of the end-to-end approach for recognizing HMEs .....	14
Figure 4. Online and Offline HME patterns .....	15
Figure 5. Summary of feature types. ....	24
Figure 6. The directional feature extraction process .....	26
Figure 7. Extracting directional feature .....	26
Figure 8. 2D space divided into nine regions of bag-of-relations .....	30
Figure 9. The relational feature vector of the “a” component .....	31
Figure 10. Selecting the number of bins for bag-of-positions .....	32
Figure 11. HSC features from Class Activation Map of class ‘i’ .....	38
Figure 12. Multi-scale aggregated HSC features for clustering. ....	38
Figure 13. Overall of the proposed network.....	39
Figure 14. Multi-scale CNN for extracting CAMs.....	40
Figure 15. Global attentive pooling (GAtP) .....	42
Figure 16. Incremental refinement approach.....	47
Figure 17. The process to synthesize an HME .....	48
Figure 18. Samples in Dset_Mix. (PQ 1: If a quadratic function $ax^2 + bx + c = 0$ has two roots, what is the general form of the roots?) .....	49
Figure 19. Purity and marking cost on Dset_Mix.....	53
Figure 20. Purity and marking cost on PQ 1: If a quadratic function $ax^2 + bx + c = 0$ has two roots, what is the general form of the roots? (the true number of answer classes is 4) .....	53
Figure 21. Purity and marking cost on PQ 7: Consider the circle, whose center is at (0,0) and radius equals $r$ . Let $(x,y)$ be a point on the circle such as the $OP$ makes an angle $\theta$ with the $Ox$ axis. Represent $x$ in terms of $r$ and $\theta$ . (the true number of answer classes is 4) .....	53
Figure 22. Visualization of clustering on question 1 (10 classes, 10 clusters).....	54
Figure 23. CAMs by multi-scale CNN with attentive pooling. The columns from left to right are the aggregated of multi-scale CAMs with class label and predictive score, the first, the second and the third level CAMs, respectively. The first row shows the combination CAMs results for all the classes.....	59
Figure 24. The pipeline of the synthesizer.....	61
Figure 25. An example of generating the root symbol with the inside sub-expression.....	66
Figure 26. Four examples of synthetic patterns in the experiments. ....	67
Figure 27. Five common unnatural reasons for synthetic patterns.....	69
Figure 28. Two UIs for showing and editing HMEs recognition result. ....	71
Figure 29. Gestures for correcting symbol segmentation errors.....	73
Figure 30. Symbol list for editing symbol misrecognition by gestures.....	74
Figure 31. Effective region of base symbol “x” enclosed by the dashed blue rectangle.....	74
Figure 32. Recognition-based gesture for editing structure analysis.....	75
Figure 33. Non-recognition-based gesture for editing structure analysis.....	76
Figure 34. Sample pattern for evaluating a gesture to edit structural relation.....	76

## Table of Tables

Table 1. Public datasets and their properties .....	21
Table 2. Rules for combining connected components .....	29
Table 3. Relations corresponding to regions .....	30
Table 4. Network configuration .....	40
Table 5. Specifications of Dset_22Qs and Dset_50 .....	48
Table 6. Dset_Mix dataset .....	49
Table 7. Purities of individual features (mean $\pm$ standard deviation) .....	50
Table 8. Purities of individual and combined features (mean $\pm$ standard deviation) .....	52
Table 9. Clustering results of HSC feature compared with other methods (purity) .....	55
Table 10. Clustering results of different configurations of multi-level spatial pooling (purity) ..	56
Table 11. Effect of global pooling methods for clustering (purity) on CROHME 2019 .....	58
Table 12. Effect of multi-scale HSC feature (purity) .....	58
Table 13. 12 Groups of symbols in CROHME 2016 dataset .....	63
Table 14. Size and position of the child symbol based on the parent symbol .....	64
Table 15. Number of each kind of patterns in the experiments .....	67
Table 16. Mean scores and Standard deviation for each kind of patterns .....	68
Table 17. How do you rank the effectiveness (Can you edit recognition errors as you like)? ....	79
Table 18. How efficiently can you edit math expressions (How quickly, easily, and so on)? ....	79
Table 19. Do you feel that your interaction with the user interface is clear and understandable? .....	79
Table 20. Do you feel it is easy to learn to use the user interface? .....	80
Table 21. How are you satisfied with the user interface? .....	80
Table 22. Which do you prefer among the recognize/edit math input, Math editor, and Math language for inputting math formula on the computer? .....	80



# **1 Introduction**

## **1.1 Background**

Nowadays, applying technology to education is becoming more and more popular. Many schools and universities have used computers and mobile/wireless devices for supporting the teaching and learning process. Especially, it is becoming a promising solution when the number of students in schools is huge and increasing every year. If there exists a system which can help teachers to mark students' work such as assignments, examinations, etc. it would be appreciated.

Among mobile devices, tablets have become popular since they provide a natural way of interaction via touch-based or pen-based interfaces. Particularly, the pen-based interfaces allow students and teachers to write notes, draw diagrams, and solving mathematical questions naturally. Especially for mathematics, a pen-based interface will make students and teachers who are not familiar with mathematical script languages (e.g. LaTeX) comfortable and save them a lot of time when inputting MEs to a computer.

The most critical component of a system for doing math on pen-based devices is the HME recognizer. Moreover, users usually demand high performance from recognizers. Anthony et al. [1] estimated that a mathematics-oriented intelligent tutoring system requires an accuracy of 91-97%. Although there are a lot of advancements in the handwriting recognition problem, the performance is still below that threshold for applying in education based on the results of international competitions on the recognition of HMEs.

Instead of applying an automatic marking mechanism that is unreliable, we can utilize a computer-assisted marking system. This approach can help teachers to reduce the time of marking mathematical answers and provide a more reliable mechanism for verifying the result than the conventional methods. Moreover, the number of markers can be reduced, and hence it makes the marking more consistent.

## **1.2 Contributions**

First, we proposed a method for clustering offline HMEs. Concretely, our method extracts multiple feature types according to multiple levels of recognition of HME images to capture rich information of MEs such as directions, symbols, relations, positions, etc. Then we combine them in order to create feature vectors and applied clustering on top of that. Moreover, we proposed a

marking cost function for evaluating the clustering methods' performance in assisting teachers to mark the mathematical answers.

Secondly, we proposed a method for generating HMEs from LaTeX strings. Currently, the problem of clustering HME has not been studied widely since there is no available dataset publicly. In addition, the data is the basic difficulty in many natural language processing (NLP) problems so we generated a synthetic dataset for encouraging research on this topic. Our method generates synthetic HMEs from LaTeX strings and handwritten patterns from CROHME 2016 which is published for the HME recognition problem. Then we can evaluate our methods on the dataset and compare it with other research in the future.

Finally, we introduce an interactive user interface for inputting MEs to computers. Currently, there are some examinations organized online on the internet and it may become popular in the future because of its flexibility and convenience, especially in the circumstance of epidemics. We designed an interactive user interface that allows users to input MEs with a pen-based screen, identify misrecognition easily, and correct the misrecognition immediately. This user interface can be applied when establishing a mathematical examination online without requiring all students familiar with programming languages (e.g. LaTeX).

### **1.3 Structures of the thesis**

The remainder of this thesis is organized in a sequential way starting with the surveys of related problems to the above topics in chapter 2. We categorize and discuss various methods in detail to give an overall view of the ME clustering problem. We also discuss the current situation of the HME recognition problem and the need for a user interface to input MEs to computers.

Chapter 3 presents our works on the topic of the offline HME clustering problem. We describe the formalization and architecture of our system including feature extraction, feature combination, clustering method, experiment results, and evaluation. In addition, we introduce a marking cost function which we used as the measurement for evaluating the performance of clustering.

Chapter 4 describes the method of generating a synthetic dataset for evaluating HME clustering methods. This dataset is published for promoting research on this topic. Other researchers can utilize the dataset for validating their methods' performance and compare with ours.

Chapter 5 introduces two user interfaces for entering MEs to computers. We present how our user interfaces show the recognition result according to multiple levels of recognition and enable the user to identify the misrecognition easily. To allow the users to correct the misrecognition, we designed a set of gestures which are efficiently editing MEs. We conducted a user study to evaluate the interfaces.

Chapter 6 and chapter 7 summarize the thesis and discuss future works based on the current results. We give some conclusions about the previous works and propose some approaches to apply our methods in practice.

## 2 Surveys

### 2.1 Offline HME Clustering

#### 2.1.1 HME Recognition (HMER)

The recognition of HME has been researched since a long time ago [2] and now it is still an active field in handwriting recognition. Given the importance of handwriting in human activities, the possibility of communicating via writing provides a major enhancement to human-machine interfaces. This allows people to communicate with machines using their natural communication skills efficiently.

Among dozens of handwriting recognition problems, HMER not only shares some common challenges such as ambiguous handwriting input, variant scales of symbols, etc. but also characterized by the complicated two-dimensional structures. For usually handwritten language recognition (e.g. English), the handwritten characters are arranged sequentially from left to right according to each line, while for HME recognition, the handwritten characters are arranged on a two-dimensional layout according to relations among them (e.g. above, below, horizontal, fraction, inside). Hence, HMER is a challenging problem.

Generally, recognition of a mathematical expression involves three tasks: symbol segmentation, symbol recognition, and structure analysis. Figure 1 shows a common architecture for HMER showing its key modules. The symbol segmentation produces symbol hypotheses (stroke partitions) - each of which is expected to form a symbol. The symbol recognition proposes a list of symbols for each symbol hypothesis. The structure analysis identifies the relations among symbol hypotheses and forms the ME interpretations (e.g. in terms of LaTeX). Each task has its challenging problems. In practice, these tasks could be processed sequentially, or simultaneously.

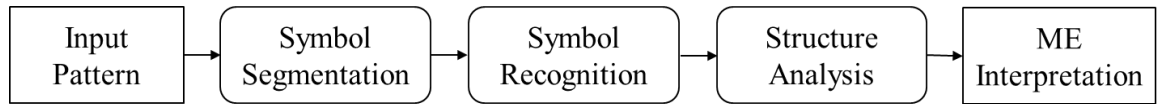


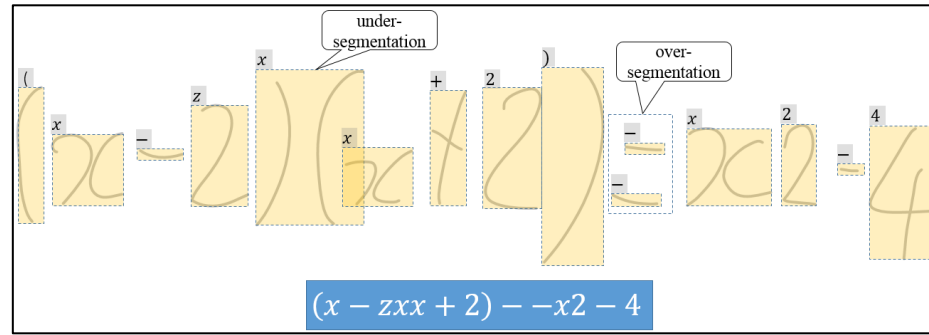
Figure 1. The flow of the general approach for recognizing HMEs

Symbol segmentation makes hypotheses for separating strokes in a given HME. The errors in the segmentation step may result in misrecognitions in the symbol recognition step. There are two main error types in the symbol segmentation step. They are under-segmentation (division into too few segments) and over-segmentation (division into too many segments) errors. Figure 2 shows

an example that contains both segmentation errors. Besides, a segmentation error can be both under- and over-segmentation (but it rarely happens).

$$(x-2)(x+2)=x^2-4$$

The original pattern



An example of the recognition result

Figure 2. Examples of error types in the symbol segmentation task

Symbol recognition provides candidates for each segmented symbol pattern. Recognition methods can be broadly classified into four categories: structural, statistical, deep neural networks, and hybrid. Structural techniques use some qualitative measurements as features. They employ rule-based, graph-theoretic, and heuristic methods for recognition. Statistical techniques use some quantitative measurements as features and an appropriate statistical method for recognition. Each hypothetical symbol is recognized by a symbol classifier. Mathematical symbol recognition is challenging because of four reasons. First, there are more than a hundred symbols such as alphanumeric, Roman symbols, Greek symbols, operators, commas, and dots. Second, many symbols are similar, such as  $[0, O, o]$ ,  $[S, s, \int]$ ,  $[\mu, u]$  and  $[\omega, w]$ , making it hard to disambiguate between them. Third, these symbols are written in different styles, typefaces, font sizes, and stroke orders. For example, the letter “a” or “a” are the same symbol, and either of the strokes “-” or “|” can be written first while composing “+”.

Finally, structural analysis is to interpret the spatial arrangements of symbols in an HME and obtain a contextually valid ME. It first determines the kind of relations between a pair of symbols

or sub-expressions. Those relations are often classified into six classes: horizontal adjacent, subscript, superscript, over, under, and inside. However, an HME may contain many ambiguities that need contextual information to find the correct structure of the expression.

According to Competitions on Recognition of Handwritten Mathematical Expressions (CROHME), the symbol segmentation and symbol recognition have achieved high performances (over 98% correct segmentation for the symbol segmentation task, 95% recognition rates for symbol recognition) [3] [4] [5] [6] [7] [8]. However, the recognition of the whole HME of the methods based on the general approach can obtain performances of only slightly over 60%.

Currently, the end-to-end approach is considered state-of-the-art in the HMER problem according to the latest competitions CROHME 2019 [9], OffRaSHME 2020 [10]. It usually employs neural networks to recognize input HMEs. The flow is shown in Figure 3 which is different but each task can be matched with a specific task in the general approach. Particularly, symbol segmentation is processed through the attention mechanism; symbol recognition is performed by decoding the focused parts in an input HME; relation classification is made based on the attention model that guides the decoder precisely attend to the direction between the current predicted symbol and the next predicted symbol. Even though the best recognition rate is reported to be approximately 80%, it is still below the expected threshold estimated by Anthony et. al. [1] to employ in a mathematics-oriented intelligent tutoring system. Hence, a computer-assisted marking system would be appreciated while the HMER systems are still unreliable.

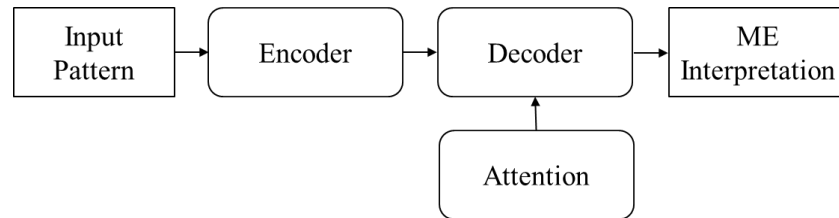


Figure 3. The flow of the end-to-end approach for recognizing HMEs

### 2.1.2 Online and Offline HMER

Before discussing the methods, I would like to clarify the types of input data of HMER. There are two types: online and offline. The online pattern is a series of coordinates of pen trajectories to write texts whereas the offline pattern is the image of the text (Figure 4). Since the nature of the data is different, the HMER is also divided into online and offline methods.

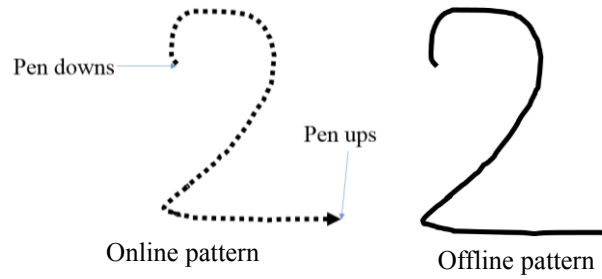


Figure 4. Online and Offline HME patterns

Depending on the input data, an HMER system may need some additional tasks, such as pre-processing, normalization, noise reduction, training language models, and so on. Online methods use resampling and smoothing, while offline methods use connected component analysis and geometric feature extraction to pre-process raw input patterns. Normalization is often applied to make the later processing easier such as symbol's size, center normalization, writing speed normalization, and size normalization for online recognition, as well as intensity normalization and moment normalization for offline recognition. Noise reduction is also commonly applied to improve input quality. In both online and offline recognition, however, the input may contain many small symbols (dot, comma, and diacritical marks), which are difficult to distinguish from noise.

### 2.1.3 Computer-Assisted Marking Systems

Examinations play an important role in education. They help teachers to evaluate learners' understanding and abilities to solve problems in order to prepare for the next stage of teaching. They also motivate learners and guides further learning. However, marking exams takes a large amount of time and effort. If it takes time to return marking results, the effect of feedback for the learners decreases. Nowadays, school teachers need to mark large quantities of student assignments and exams.

Mark sheets and Computer/Web-based testing have been introduced to solve this problem. Due to the current level of IT, however, they often restrict the questions to selection types. Descriptive questions have been generally excluded in those IT-empowered examinations. The time and cost to mark answers are reduced and marking errors are lessened but they sacrifice to measure learners' deep understanding and their ability to think. Even if an examinee does not

understand a question or cannot answer it, he/she can get a correct mark by selecting a correct answer through guessing. It has also the bad side effect for learners to “select” rather to “think”.

Automatic marking is a solution to the exam marking problem. Extensive research has been carried out on essay assessment [11], [12], [13], and even handwritten essay scoring was reported [14] for English examinations, but it does not deal with the correctness of answers. Plenty of research has been made on auto-grading of programming [15], [16], [17], but automatic marking is mostly unable to handle partially correct answers. Among others, the most practical problem of automatic marking is that it cannot be used without learners’ confirmation of marking. Large-scale examinations such as national-wide qualification examinations and entrance examinations do not provide such opportunities for examinees to confirm marking so that the marking of those examinations cannot be automated beyond selection types.

Another solution is computer-assisted marking (CAM). It may provide efficiency and even reliability for marking. One of the most promising approaches is to cluster answers. If answers are well clustered, they can be marked efficiently, and marking errors and fluctuations will be decreased. Specifically, markers can mark multiple similar answers in a cluster by just one action instead of marking one by one as the conventional method. Since the final marking is made by human markers, examinees’ anxieties will be also reduced. For large-scale examinations whose marking must be made in a short period without errors, computer-assisted marking is a promising solution. For daily exams as well, teachers’ burden will be reduced and more time will be devoted to giving feedback to learners.

S. Basu et. al. proposed a clustering approach for English short answer grading [18]. Their approach uses k-means to group student responses into clusters and subclusters, then they automatically mark clusters and subclusters using predefined answer keywords. Finally, the system allows teachers to read, mark, and provide feedback on those groups of answers at once.

M. Brooks et. al. proved that clustered interface is superior in terms of ease of use and overall effectiveness [19]. However, there are only a few systems focused on handwritten patterns. Moreover, the number of existing systems for mathematical examinations is even fewer.

As far as we know, there is no research about assisting in marking handwritten math answers currently. The most reason is due to the ambiguities, complicated two-dimensional structures, and multiple equivalent representation forms of math expressions. The CAM groups a large number of answers into a smaller number of groups and let the teachers mark each group of answers by



one action. Hence, the workload of teachers is significantly reduced, thus fewer teachers are employed for marking. Students' anxieties for marking will be reduced since the final marking is manually made by teachers. Hence, the clustering-based CAM for HME answers is expected to make the marking process more efficient and decrease marking errors as well as reduce score fluctuations compared to completely manual marking.

#### **2.1.4 Survey on Clustering Algorithms**

In general, the traditional clustering algorithms can be divided into 9 categories [20] based on these approaches: partition, hierarchy, distribution, density, fuzzy theory, graph theory, grid, fractal theory, and model. However, the most commonly used ones are the first four ones.

##### ***2.1.4.1 Partitional Clustering***

In the clustering algorithms based on the partition, the data is divided into several non-overlapping subsets such that each data instance is assigned to exactly one subset. Because checking all possible partition is computationally infeasible, heuristics are used in the form of iterative optimization. Specifically, by relocation schemes that iteratively reassign points between the clusters, the algorithms gradually improve clusters' quality.

For example, k-means [21] is a typical partitioning method that applies an iterative refinement approach with two main steps. The first step is to choose the means of clusters as the centroids, whereas the second step is to assign data points to their nearest centroids. In practice, its computational speed and simplicity appeal to people [22], [23]. Its main drawback is the vulnerability to its random seeding technique. In other words, if the initial seeding positions are not chosen correctly, the clustering result quality will be affected adversely. In light of that, David Arthur and Sergei Vassilvitskii proposed a method called k-means++ [24] to improve k-means. The main difference between k-means and k-means++ is the seeding technique. A new seeding technique is proposed to replace the arbitrary seeding technique of k-mean. Given a set of seeds chosen, the seeding technique favors the data points which are far from the seeds already chosen. Thus the seeds are chosen probabilistically as dispersed as possible.

Some other variants of k-means are k-medoids [25], in which the basic idea is to update the center of data points as the center of the corresponding cluster until some criteria of convergence are satisfied. Besides, PAM [26], CLARA [27] are also typical examples of this approach.

The advantage of this clustering approach is relatively low time complexity and high computing efficiency in general. However, this disadvantage is only suitable for convex

clustering. Moreover, it is relatively sensitive to the outliers, easily drawn into local optimal and a major disadvantage is the number of clusters needed to be predefined and the clustering result sensitive to the number of clusters.

#### 2.1.4.2 Hierarchical Clustering

Hierarchical clustering builds a hierarchical structure for clustering the data. That structure is presented as a tree of clusters which are also known as a dendrogram. Every cluster node contains child clusters. Sibling clusters partition the points covered by their common parent. This approach allows the data can be partitioned into different levels based on some criteria.

Hierarchical clustering methods are categorized into bottom-up (agglomerative) and top-down (divisive) [26]. A bottom-up clustering starts with one-point (singleton) clusters and recursively merges two or more most appropriate clusters. A top-down clustering starts with one cluster of all data points and recursively splits the most appropriate cluster. The process continues until a stopping criterion is achieved.

Single-linkage clustering [28] is one of the most popular hierarchical clustering methods. It is a classic bottom-up approach in which data points are gradually agglomerated together to form clusters. In each step, all pair-wise distances are computed to identify the minimum. The parties involved in the minimal pair-wise distance are linked together. Such a step is repeated until all data points are linked together. A hierarchical tree is constructed to connect all data points at the end. A tree depth level can be chosen to cut the tree, forming clusters. distribution-based clustering.

The main advantage of hierarchical clustering is the flexibility of the level of the clustering result because the hierarchical relationship among clusters easily detected, and relatively high scalability in general. In contrast, the disadvantage of most hierarchical algorithms is they do not revisit once constructed clusters so we cannot turn back to improve the previous cluster level.

#### 2.1.4.3 Distribution-Based Clustering

In the distribution-based clustering, the data is considered to be a sample independently drawn from a mixture model of several probability distributions. The data belong to the same cluster, have the same distribution. Concretely, the data points are generated by, first randomly picking a model  $M_i$  with probability  $p_i$  in ( $i = 1, \dots, n$  where  $n$  is the number of models) and then drawing a point from the corresponding distribution. Therefore, we can say a cluster corresponds to a set of distribution parameters such as mean, variance, etc.

The typical algorithms in this approach are the Gaussian mixture model [29] and DBCLASD (Distribution-Based Clustering of LARge Spatial Databases) [30]. The common idea underlying these algorithms is to estimate the parameters of the distributions. There one more thing is the mixture model has to predefine the number of distribution models, the determination of the most suitable number of clusters is required.

This clustering approach is more realistic to give the probability of belonging, relatively high scalability by changing the distribution, the number of clusters, and so on, and supported by the well-developed statistical science. However, it strongly depends on the assumption of the data distribution which is involved in many parameters and relatively high time complexity.

#### 2.1.4.4 *Density-Based Clustering*

The basic idea of this kind of clustering algorithms is that the data which is in the region with a high density of the data space is considered to belong to the same cluster. The interesting thing is the clustering algorithm based on this approach could handle outliers. The typical algorithms include DBSCAN (Density-based spatial clustering of applications with noise) [31], OPTICS (Ordering Points To Identify the Clustering Structure) [32], etc.

DBSCAN is the most well-known density-based clustering algorithm, which is generated from the basic idea of density-based clustering algorithms directly. The crucial concepts of DBSCAN are density and connectivity both measured in terms of local distribution of nearest neighbors. Two input parameters  $\varepsilon$  and MinPts are used to define:

1. The  $\varepsilon$ -neighborhood of the point  $x$  is  $N_\varepsilon(x) = \{y \in X | d(x, y) \leq \varepsilon\}$
2. The core object: a point with a neighborhood consisting of more than MinPts points
3. The concept of a point  $y$  density-reachable from a core object  $x$ : a finite sequence of core objects between  $x$  and  $y$  exists such that each next belongs to an  $\varepsilon$ -neighborhood of its predecessor
4. A density-connectivity of two points  $x, y$ : they should be density-reachable from a common core object

So the defined density-connectivity is a symmetric relation and all the points reachable from core objects can be considered as clusters. The points that are not connected to any core point are declared to be outliers (they are not covered by any cluster). The non-core points inside a cluster represent its boundary. Finally, core objects are internal points. Processing is independent of data ordering.

The algorithm OPTICS is an improvement of DBSCAN and it overcomes the shortcoming of DBSCAN that being sensitive to two parameters,  $\varepsilon$ , and MinPts. Although keeping the same two parameters, OPTICS builds an ordering of data that is not only consistent with DBSCAN but also includes more information about the distance between data points. With each point, OPTICS defines two additional distance, the core-distance, and reachability-distance. The core-distance is the distance to MinPts' nearest neighbor when it does not exceeds  $\varepsilon$ , or undefined otherwise.

The OPTICS can be considered as a DBSCAN extension since instead of relying on user-defined parameters, the OPTICS could explore the data probability distribution without  $\varepsilon$ . By setting the  $\varepsilon = +\infty$ , the distance from a point to its nearest neighbor is always defined. Hence, each cluster has its typical distance-to-nearest-neighbor scale.

The density-based clustering algorithms are efficient and suitable for data with arbitrary shapes. However, when the density of the data space is not even, the clustering result is sensitive to the parameters.

## **2.2 Synthetic Data Generation**

### **2.2.1 Datasets**

The public datasets are useful to evaluate and compare the performance of HME recognition systems. Almost datasets are online, except MNIST, CROHME 2019, and OffRaSHME 2020 provide offline patterns (image). The offline datasets are mainly rendered from online patterns due to the lack of real images of HME. However, there are some differences between the true offline patterns and the simulated offline patterns:

- True offline patterns usually contain uneven stroke width, grayscales, heavy noise, and sometimes show-through.
- The writing feeling of the pen is different on a digital device than on paper, which may change the writing style of the same person.

Another approach that is different from the rendering online patterns approach is synthesizing new patterns from the old ones to generate new patterns. This approach is usually used for augmenting data by utilizing available patterns [33] and applying some transformation techniques to generate new data.

### **2.2.2 Measurements for HMER**

Evaluation methods of HME recognizers are usually based on comparing the recognition results with the ground truths. A more thorough evaluation, however, needs to consider different

stages of the process, and not only the final results, because an error in recognizing a base-level structure is more severe than an error in recognizing expressions at a higher level.

Table 1. Public datasets and their properties

Data set	Type	Amount	Format
MNIST [34]	Digits	10 classes	Gray-scale image Size normalized (28x28)
Brown [35]	Symbols	48 classes	Unipen format
MathBrush [36]	Expressions	Total: 4655 samples	Microsoft and SCG ink
Chan and Yeung [37]	Expressions	Total: 600 samples	CRC standard mathematical Tables and Formula [38]
CIEL [39]	Symbols	34 classes	InkML format
	Expressions	Train: 6480 samples Test: 3600 samples	
HAMEX [40]	Expressions	Train: 2925 samples Test: 1425 samples	InkML format
MfrDB [41]	Expressions	Total: 2018 samples	InkML format
ExpressMatch [42]	Expressions	Total: 910 samples	InkML format
CROHME 2011 [43]	Symbols	56 classes	InkML format
	Expressions	Train: 921 samples Test: 348 samples	
CROHME 2012 [44]	Symbols	75 classes	InkML format
	Expressions	Train: 1341 samples Test: 486 samples	
CROHME 2013 [6]	Symbols	101 classes	InkML format
	Expressions	Train: 8836 samples Test: 671 samples	
CROHME 2014 [45]	Symbols	101 classes + junk	InkML format
	Expressions	Train: 8836 samples Test: 986 samples	
	Matrices	Train: 362 samples Test: 175 samples	
CROHME 2016 [3]	Symbols	101 classes + junk	InkML format
	Expressions	Train: 8836 samples Valid: 986 samples Test: 1147 samples	
	Matrices	Train: 362 samples Valid: 175 samples Test: 250 samples	
CROHME 2019 [9]	Symbols	101 classes + junk	InkML format and Image
	Expressions	Train: 9993 samples Valid: 986 samples Test: 1199 samples	InkML format Image
	Document Page	Train: 569 pages, 26395 ME regions Test: 236 pages, 11885 ME regions	Image

Therefore, graph-based metrics have been proposed since the structure of an ME can be represented by a tree structure. Overall the correctness of an expression is measured by counting the number of operators for correcting the graph of the recognition result to match with that of the ground truth. The operators are usually the ones for modifying nodes and edges [46] [47] [48] [9]. These metrics provide global measures that take into account a specific part of the recognition results as they provide information about the recognition errors.

### **2.3 User Interface for Inputting MEs**

Currently, there are three primary methods for entering mathematical expressions (MEs) into computers, by mathematical script languages, by mathematical formula editors, and by computer recognition of handwritten mathematical expressions (HMEs). The mathematical script languages require the users to be familiar with a programming language (e.g., LaTeX) to input MEs. Meanwhile, the mathematical formula editor (e.g., MathType) often forces the users to select predefined mathematical structure templates (e.g., fractions, superscripts, subscripts) from a menu or toolbar and then fill in the templates with numbers, letters, and so forth by typing on the keyboard. These two methods, however, have difficulties for students to use in the learning process, especially in answering descriptive questions.

On the other hand, the HME recognition-based method is a natural choice for inputting MEs. However, this method requires a highly reliable HMEs recognizer. HMEs are more challenging to recognize than natural languages since the structural ambiguity is higher. In fact, the recognizer of HMEs usually has problems with recognition errors. The errors could come from various levels of recognition: symbol segmentation, symbol recognition, and structure analysis. By providing a user interface with which the user can quickly correct recognition errors, the user experience could be significantly improved.

For handwritten Japanese, there are some similar works such as [49], [50]. For mathematics, Steve Smithies et al. proposed a prototype math editor that incorporated a set of simple procedures for correcting errors made in recognition [51]. In 2016, Taranta et al. introduced a dynamic system that used Math boxes for writing complicated mathematical expressions [52]. Besides, they incorporated some editing features which allowed the users to edit HMEs before and after their recognition and correct recognition errors. In the same year, Tokuda et al. also proposed an error correction user interface for inputting mathematical expressions by handwriting recognition [53].

### **3 Offline HME Clustering**

#### **3.1 Introduction**

In this chapter, we present the clustering handwritten mathematical expressions for computer-assisted marking since mathematical expressions are typical descriptive answers. Examinees' understanding and problem-solving abilities are most clearly revealed.

Many approaches enable teachers to digitalize students' answers and mark them on the computer. However, they are still limited for supporting marking descriptive mathematical answers that can best evaluate learners' understanding. This chapter presents the clustering of offline handwritten mathematical expressions to help teachers efficiently mark answers in the form of HMEs.

In this work, we investigate a method of combining feature types from low-level directional features and multiple levels of recognition: bag-of-symbols, bag-of-relations, and bag-of-positions. Moreover, we propose a marking cost function to measure the marking effort. To show the effectiveness of our method, we used two datasets and another sampled from CROHME 2016 with synthesized patterns to prepare correct answers and incorrect answers for each question. In experiments, we employed the k-means++ algorithm for each level of features and considered their combination to produce better performance. The experiments show that the best combination of all the feature types can reduce the marking cost to about 0.6 by setting the number of answer clusters appropriately compared with the manual one-by-one marking.

#### **3.2 Our Proposed Method**

The clustering of handwritten MEs is one of the current challenges concerning handwriting recognition because an ME could contain many kinds of symbols such as Roman symbols, Arabic numerals, Greek symbols, operators, fraction lines, root, commas, dots, and so on. Moreover, these expressions can be written in different sizes and/or different styles. The feature extraction process is challenging because each writer has his writing habit which is different from the others. We need a feature that invariant to the writing styles.

We focus on clustering handwritten mathematical expressions for computer-assisted marking, as mathematical expressions are typical descriptive answers, which reveal the students' understanding and problem-solving abilities. Here, however, we must solve several problems to make computer-assisted marking effective. First, we must extract features from HMEs without entirely depending on their computer recognition. HME recognition is one of the most difficult

handwriting recognition problems, with the best performance being around 80% for the CROHME 2019 dataset [9]. Secondly, we need a measure to evaluate how “well” a clustering method supports markers in the marking process, and to compare the performance of different methods. Thirdly, we must consider the user interaction between a marker and a marking system as clustering is not perfect. Due to many ambiguities in HMEs, the marker should be able to verify and regrade the “impure” elements efficiently. Fourthly, we must find the best clustering algorithm for this purpose. Fifthly, a lack of public datasets limits the development of research in this approach.

### 3.3 Multi-Level Features From an HME

Low-level pattern features can be employed for clustering. They are free from HME recognition accuracy but they are not robust to the various ways to write an HME. Features produced from several levels of recognition, such as recognition candidates of symbols produce useful distinctive information although recognition results are fragile due to the immaturity of HME recognition.

A promising solution to this problem is to extract features from HMEs at multiple levels of recognition: low-level directional features, bag-of-symbols, bag-of-relations, and bag-of-positions. Figure 5 shows these features according to their levels. The low-level features are robustly extracted from an HME image but they are often sensitive to the way it is written.

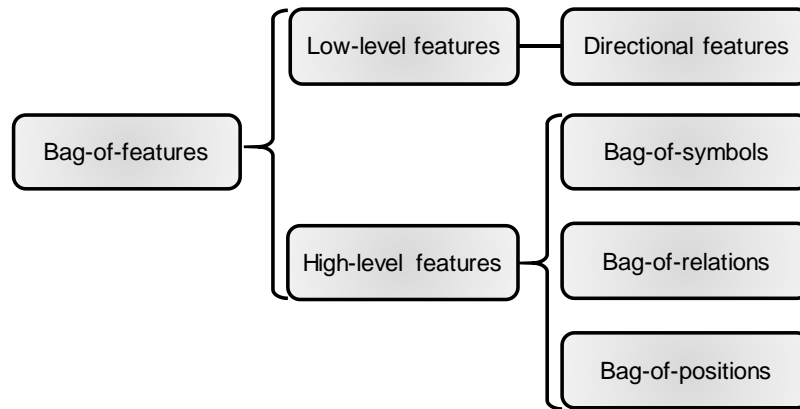


Figure 5. Summary of feature types.

As for the high-level features, we extract symbols, recognize them, represent their positions and relations in the form of bag-of-symbols, bag-of-relations, and bag-of-positions, without



applying the HME recognition yet. This is because the HME recognition applies structural analysis and total optimization in the context of a mathematical expression grammar on top of symbol segmentation and recognition so that the entire process is often fragile... Therefore, we avoid the last stage of structural analysis and total optimization. A convolutional neural network (CNN) is applied for recognizing symbols. These bag-of-features are considered as high-level features as they are extracted from symbols recognized from an HME. The high-level features are dependent on symbol recognition but are robust to the way an HME is written in different styles.

In the experiments, we evaluate the effect of the low-level and high-level features and consider combining them to improve clustering performance. The rest of this section describes these features in detail and present the combination strategies.

### 3.3.1 Directional features

Directional feature extraction has been established for offline character images [54] [55], there are four main steps in the process: nonlinear normalization, directional decomposition, Gaussian blurring, and sampling. Firstly, the HME image is normalized to the size of  $64 \times 64$  by using nonlinear normalization. Secondly, eight-direction planes are used corresponding to eight chain code directions to compute the magnitude and direction of gradients in the normalized image. Then, each plane is divided into  $8 \times 8$  regions. A low-pass Gaussian filter is applied for all the regions with some overlap to enhance the robustness of positional distortions. Finally, all the directional magnitudes of all the regions and all the planes are concatenated into a single vector.

In this work, we normalize the size of HMEs to the average size of the expressions for the answer instead of normalizing them to a fixed size such as  $64 \times 64$ . Assume the size is the height  $H \times$  the width  $W$ . Then, we decide how many partitions to apply. For a given image of the size  $H \times W$ , we calculate the average height and width of the connected components within the image (denoted as  $\bar{H}_{comp}$  and  $\bar{W}_{comp}$  respectively), then we calculate the number of regions  $M \times N$  as shown in (1).

$$M = \frac{H}{\bar{H}_{comp}}, N = \frac{W}{\bar{W}_{comp}} \quad (1)$$

The extraction progress of the directional features includes seven steps as in Figure 6. Figure 7 illustrates the process briefly. After reading the input image, the region of interest (ROI) containing mathematical expressions (MEs) is obtained by scanning the boundary. The size of ROIs is not constant, depend on the type of MEs and the user's writing style.

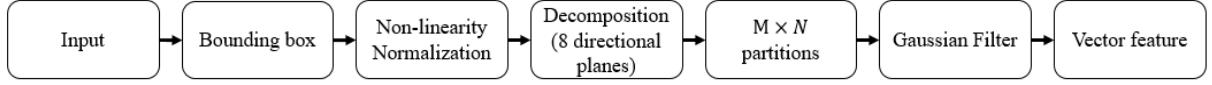


Figure 6. The directional feature extraction process

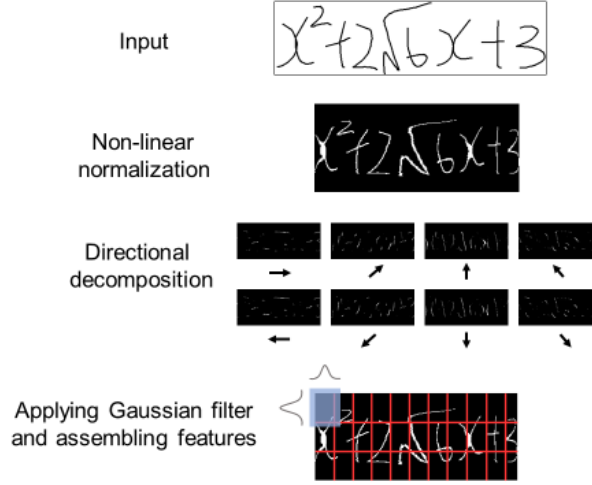


Figure 7. Extracting directional feature

Then, the line density projection interpolation (LDPI) method is applied for the ROI. This step aims to regularize the size, position, and shape of the ROI to reduce the shape variation between the samples of the same group. Furthermore, this normalization maps ROI onto a standard image plane so that it has the same dimensionality. The size of the normalized image could be set to a fixed size or the average size of ROI of all samples in our database.

An original sample  $I(x, y)$  is transformed into a normalized sample  $I'(x', y')$  by coordinate mapping of pixels:

$$\begin{cases} x' = u(x, y) \\ y' = v(x, y) \end{cases} \quad (2)$$

Based on [56], the coordinate mapping function  $u(x, y)$  is computed according to pseudo-2D normalization by combining three 1D coordinate functions  $u_i(x)$  with three weight functions  $w_{i,u}(y)$  for  $i = 1, 2, 3$ . Using three soft strips of  $I(x, y)$ , the function is obtained as in (3).

$$u(x, y) = \begin{cases} w_{1,u}(y)u_1(x) + w_{2,u}(y)u_2(x), & y < y_c \\ w_{3,u}(y)u_3(x) + w_{2,u}(y)u_2(x), & y \geq y_c \end{cases} \quad (3)$$

where  $(x_c, y_c)$  is the centroid of  $I(x, y)$ .

Similarly, we also could have  $v(x, y)$ .

$$v(x, y) = \begin{cases} w_{1,v}(y)v_1(x) + v_{2,v}(y)u_2(x), & x < x_c \\ w_{3,v}(y)v_3(x) + v_{2,v}(y)u_2(x), & x \geq x_c \end{cases} \quad (4)$$

Next, eight directional decomposition planes are extracted directly from the normalized image. By using the Sobel operator, the gradient vector is decomposed into components in eight chain code directions. The gradient vector  $g(x, y) = [g_x, g_y]^T$  at the pixel  $(x, y)$  in the normalized image is computed by (5).

$$\begin{aligned} g_x(x, y) &= f(x+1, y-1) + 2f(x+1, y) + f(x+1, y+1) \\ &\quad - f(x-1, y-1) - 2f(x-1, y) - f(x-1, y+1) \\ g_y(x, y) &= f(x-1, y+1) + 2f(x, y+1) + f(x+1, y+1) \\ &\quad - f(x-1, y-1) - 2f(x, y-1) - f(x+1, y-1) \end{aligned} \quad (5)$$

The gradient's magnitude and direction can be computed from the vector  $[g_x, g_y]^T$ . The length of the vector of each component represents the intensity of gradient in the corresponding direction plane. The vector is assigned to the corresponding direction plane. The image shows an example of 8 directional planes extracted from the normalized image in the previous plane.

The value of the pixels on each directional plane is sampled uniformly according to the Sampling Theorem. Before sampling, we divide each plane into  $D_{ver} \times D_{hor}$  partitions, where  $D_{ver}$  and  $D_{hor}$  is the number of divided partitions in vertical and horizontal respectively. These two parameters are set as the maximum of divided partitions on whole samples in the database based on the average size of connected components. Suppose  $I = \{I_1, I_2, \dots, I_N\}$  is the set of normalized ROIs of all samples,  $H_i$  and  $W_i$  are the height and width of  $I_i$ . For each image  $I_i$ , we calculate  $C_i^h$  and  $C_i^w$ , which is the average height and width of the connected components in the image  $I_i$  after removing outliers (fraction, dot, noise, etc.). Following computed values,  $D_{ver}$  and  $D_{hor}$  could be calculated by:

$$D_{ver} = 3 \times \max_{i=1, N} \{D_i^{ver}\}, D_{hor} = \max_{i=1, N} \{D_i^{hor}\} \quad (6)$$

where  $D_i^{ver} = H_i/C_i^h$ ,  $D_i^{hor} = W_i/C_i^w$  are the number of divided partitions of the image  $I_i$  in vertical and horizontal respectively.

Then, the regions are blurred using a low-pass Gaussian filter to make connected information between neighboring regions. The variant parameter  $\sigma_x$  of the Gaussian filter is related to the sampling interval between blurring masks  $t_x$  [57].

$$\sigma_x = \frac{\sqrt{2} t_x}{\pi} \quad (7)$$

The  $t_x$  is set as  $\frac{H+W}{2}$ , where  $H$  and  $W$  are the height and width of the divided region. As a result, we obtain  $D_{ver} \times D_{hor}$  feature values from each direction plane for the total of  $D_{ver} \times D_{hor} \times 8$  feature values. To improve the Gaussianity of feature distribution, we apply power transformation with the power parameter equaling to 0.5 [58]. Finally, the values of the feature vector are normalized into the range  $[0, 1]$ .

### 3.3.2 Bag-of-symbols

Bag-of-symbols is one of the most popular features employed for clustering. Symbols in a mathematics examination could be digits, letters, operators, fence symbols, etc. In this work, we assume an HME includes 101 mathematical symbols which are mentioned in

The symbols included:

- Digits (0-9)
- Letters (a-z, A-C, E-I, L-N, P, R-T, V, X, Y)
- Greek Letters ( $\alpha$ ,  $\beta$ ,  $\gamma$ ,  $\lambda$ ,  $\phi$ ,  $\pi$ ,  $\theta$ ,  $\sigma$ ,  $\mu$ ,  $\Delta$ )
- Arithmetic Operators (+, -,  $\pm$ ,  $\div$ ,  $!$ ,  $\times$ ,  $/$ )
- Logical Operators ( $\rightarrow$ ,  $|$ ,  $\forall$ ,  $\exists$ )
- Set Operators ( $\in$ )
- Operators with Limits ( $\Sigma$ ,  $\int$ )
- Functions and Relations ( $=$ ,  $\neq$ ,  $<$ ,  $\leq$ ,  $>$ ,  $\geq$ ,  $\log$ ,  $\sin$ ,  $\cos$ ,  $\tan$ ,  $\lim$ )
- Fence symbols ( , , { , } , [ , ]
- Other symbols ( $\infty$ , COMMA, ., ..., etc.)

Hence, bag-of-symbols is a 101-dimensional vector. Instead of using the occurrence counts of the symbols, however, we allow the features to take some “fuzzy” values. We employ a convolutional neural network (CNN) which has been achieved much success in recognizing handwritten mathematical symbols [59].

We employ a CNN architecture presented in [60]. This CNN is applied for each connected component in an HME image to obtain the vector of occurrence probabilities of the symbol categories. Then, we add all the vectors to form the feature vector of the entire HME image by (2).

$$BoS = \sum_{i=1}^{n_{CC}} BoS_{CC_i} \quad (8)$$

where  $BoS$  is the feature vector of bag-of-symbols from the entire HME image,  $BoS_{CC_i}$  is that of bag-of-symbols from the  $i^{\text{th}}$  connected component, and  $n_{CC}$  is the total number of connected components inside the HME image.

For symbols composed of multiple components, a set of rules is predefined for combining them as shown in Table 2. Each rule is associated with a threshold to reduce wrong combinations. According to the rules, if some individual components are combined and reclassified as the corresponding symbol with a higher probability than the corresponding threshold by the CNN, they are recognized as the corresponding symbol. For example, if a pair of “-” components is reclassified as the “=” symbol by the CNN with a probability higher than 0.7, these “-” components are recognized as the “=” symbol.

Table 2. Rules for combining connected components

Component 1	Component 2	Symbol	Threshold
-	-	=	0.7
<	-	≤	0.8
>	-	≥	0.8
	.	!	0.5
+	-	±	0.5

### 3.3.3 Bag-of-relations

The relationship between the components under consideration is represented by their relative positions to each other. A bag-of-relations represents how many types of relations occur in an HME. The bag-of-relations of each component counts its frequencies of relations with the neighbor components.

We divide two-dimensional space into nine regions which are originated from the component under consideration and numbered as shown in Figure 8. Each region represents a specific relation

between the components as shown in Table 3. For example, the region “0” represents the inside relation (which appears in root expressions), the region “1” represents the above relation, etc.

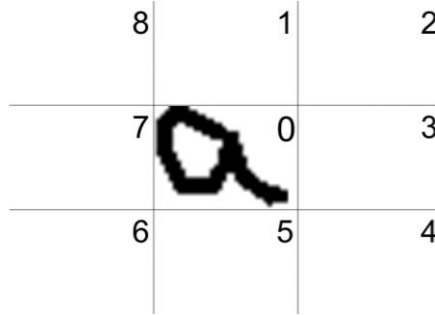


Figure 8. 2D space divided into nine regions of bag-of-relations

Table 3. Relations corresponding to regions

Region	Represented relation
0	Inside
1	Over
2	Superscript
3	Right horizontal
4	Subscript
5	Under
6	Pre-subscript
7	Left horizontal
8	Pre-superscript

For each extracted component, its neighboring components and the relationships among them are determined by xy-projections. Two components are neighbors to each other if there is no other component whose bounding box overlaps the line connecting the centers of their bounding boxes. Then, we count the number of connected components in each region based on the centers of their bounding boxes so that this feature value is discrete.

Figure 9 shows an example of extracting a bag-of-relations for the component “a”. For each component, we produce a feature vector for bag-of-relations. After extracting the bag-of-relations from all the components, we calculate their cumulative sum for forming the final feature vector of the entire HME pattern.

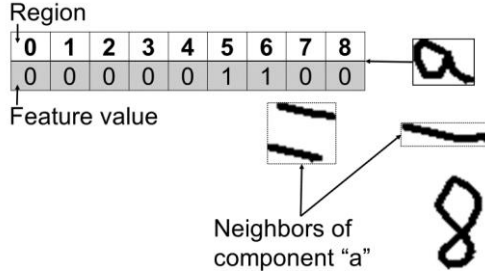


Figure 9. The relational feature vector of the “a” component

### 3.3.4 Bag-of-positions

A bag-of-positions represents where components exist in an HME image. We extract the bounding box of an input HME and divide it into uniform bins of rectangles ( $M \times N$  bins). Hence, a feature vector for bag-of-positions is an  $MN$ -dimensional vector.

In this work, the number of bins is chosen based on the structure of the true answer assuming that students write the true answer and wrong answers similar to the true answer. When students write an answer, they usually imagine a baseline to guide the following writing. Moreover, an HME often has relations of superscript, subscript, above, and below, which leads to creating sub-baselines for writing terms having these relations. The number of sub-baselines of an expression  $SB_{Exp}$  are determined by (9):

$$SB_{Exp} = \max(SB_{sup}, SB_{above}) + \max(SB_{sub}, SB_{below}) \quad (9)$$

where  $SB_{sup}$ ,  $SB_{sub}$ ,  $SB_{above}$ ,  $SB_{below}$  are the number of sub-baselines of the superscript term, that of the subscript term, that of the above term, and that of the below term, respectively.

We assume that the terms at the same level in a nested structure share the same sub-baseline. For example, in the expression “ $a^{b^c} + d^{ef}$ ”, the pairs of terms “ $a$ ” and “ $d$ ”, “ $b$ ” and “ $e$ ”, “ $c$ ” and “ $f$ ” share the same sub-baseline. The numbers of rows and columns of bins are determined by examining the baseline and sub-baselines in the answer. The number of rows is chosen to be the total number of the baseline and sub-baselines, while the number of columns is the number of symbols on the baseline plus the largest number of symbols among the sub-baselines. Figure 10 shows an example. The number of rows is three as there are one baseline and two sub-baselines for the numerator and the denominator of the fraction. The number of columns is five as there are two symbols on the baseline and three symbols (the largest number of symbols) on the sub-baselines.

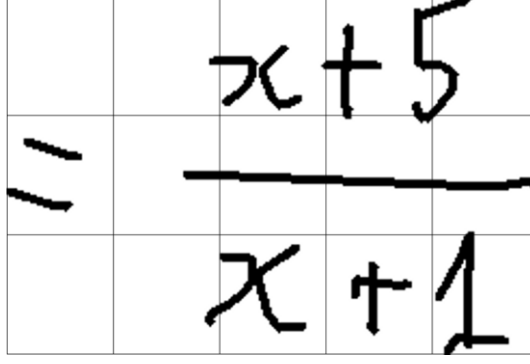


Figure 10. Selecting the number of bins for bag-of-positions

For each component, we compute the distribution of the component's pixels in each bin by (10):

$$P(G, C) = \frac{\# \text{ pixels of } C \text{ inside } G}{\# \text{ pixels of component } C} \quad (10)$$

Then, we apply a Gaussian Filter with  $\sigma_x$  and  $\sigma_y$  being  $\frac{\sqrt{2}}{\pi} \times g_x$  and  $\frac{\sqrt{2}}{\pi} \times g_y$  where  $g_x$  and  $g_y$  are the width and height of each bin. The final feature vector is the cumulative summation of all the components by (11):

$$P = \left( \sum_C P(G_{11}, C), \sum_C P(G_{12}, C), \dots, \sum_C P(G_{NM}, C) \right) \quad (11)$$

where  $G_{nm}$  ( $1 \leq n \leq N, 1 \leq m \leq M$ ) is the bin at the  $n^{th}$  row and the  $m^{th}$  column.

### 3.3.5 Feature combination methods

Combining multiple feature types is expected to form more robust features for clustering HMEs. Equation (12) presents the concatenated form of these feature types:

$$F = (F_1, F_2, \dots, F_i, \dots, F_H) \quad (12)$$

where  $F_i = (f_{i1}, f_{i2}, \dots, f_{iD_i})$  ( $1 \leq i \leq H$ ) and  $D_i$  is the dimension of  $F_i$ .

Given two data points  $p, q$  in the dataset  $Dset$ , the distance between their combined features  $F^p$  and  $F^q$  is the cumulative distance over each feature type  $F_i^p, F_i^q$  as shown in (13):

$$d(F^p, F^q) = \sum_{i=1}^K d(F_i^p, F_i^q) \quad (13)$$

where  $d(F_i^p, F_i^q)$  is the distance between the feature types  $F_i^p$  and  $F_i^q$ . We used Euclidean distance.



Then, we present three methods to combine feature types for clustering, namely normalization, standardization, and weighted combination. The first method, denoted as Normalized\_Comb, normalizes the individual feature vectors and concatenates them together. The normalized form of  $F$  is shown in (14):

$$\hat{F} = (\hat{F}_1, \hat{F}_2, \dots, \hat{F}_i, \dots, \hat{F}_H) \quad (14)$$

where  $\hat{F}_i$  is the normalized form of  $F_i$  by scaling the values of each feature type to the interval of  $[0,1]$ . Given a data point  $p$  in  $Dset$ , each individual feature type  $F_i^p$  ( $1 \leq i \leq K$ ) is normalized as shown in (15):

$$\hat{F}_i^p = \frac{F_i^p - \min_{f_{il}^x \in F_i^x, x \in Dset} f_{il}^x}{\max_{f_{il}^x \in F_i^x, x \in Dset} f_{il}^x - \min_{f_{il}^x \in F_i^x, x \in Dset} f_{il}^x} \quad (15)$$

The second method, denoted as Standardized\_Comb, makes the distribution of each feature type have zero-mean and unit variance. Given a data point  $p$  in  $Dset$ , each individual feature type  $F_i^p$  ( $1 \leq i \leq K$ ) is standardized as shown in (16):

$$\hat{F}_i^p = \left( \frac{f_{i1}^p - \mu_{i1}}{\sigma_{i1}}, \frac{f_{i2}^p - \mu_{i2}}{\sigma_{i2}}, \dots, \frac{f_{iD_i}^p - \mu_{iD_i}}{\sigma_{iD_i}} \right) \quad (16)$$

where  $\mu_{ij}$  is the mean value and  $\sigma_{ij}$  is the standard deviation of  $f_{ij}$ .

The third method, denoted as Weighted\_Comb, uses a set of weighting parameters for all the feature types to optimize clustering performance. As the clustering performance of each feature type is different, the Weighted\_Comb allows feature types to contribute to the clustering at different degrees using the weighting parameters. The feature type which contributes more effectively is weighted higher than the less effective one. A training set is used to determine the optimal weighting parameters. We use the grid-search technique to find the optimal weighting parameters.

The weighted distance is calculated by extending (13) as shown in (17):

$$d(F^p, F^q) = \sum_{i=1}^K w_i d(F_i^p, F_i^q) \quad (17)$$

where  $w_i$  is the weight trained for the distance in the feature type  $F_i$  with  $\sum_i w_i = 1$ .

### 3.3.6 Positional bag-of-features

Another approach for improving the clustering performance is extracting bag-of-features by local regions. We call it positional bag-of-features. Particularly, we divide the image into a mesh as the bag-of-positions feature. For each bin in the mesh, we extract bag-of-symbols and bag-of-relations. Then, we concatenate these bag-of-features bin by bin. The advantage of this approach is preserving the information of relative positions among symbols.

### 3.4 CNN-based Spatial Classification Features

Recently, there are some deep neural network-based approaches to cluster images such as Deep Embedded Clustering (DEC) [61], Deep Clustering Network (DCN) [62], Siamese Network [63], Deep Adaptive Clustering (DAC) [64]. These methods, however, focused on learning class discriminative features for clustering images, which are hard to generalize for unconstrained HMEs. Since HMEs have unlimited compositions of symbols and structured relations, learning embedded features to represent the spatial information of MEs is difficult. In this work, we proposed a deep neural network-based method to learn the structural representation of local features in HME images. The network learns to detect and classify the symbols in HME images and produce the symbol classification as local features and spatial information of these symbols as the structure of the local feature. The hierarchical combination of the structural representation is used for clustering HME images.

Among the deep neural network-based clustering methods, Deep Embedded Clustering (DEC) proposed by Xie et al. [61] is an early study, which has the structure of an Auto Encoder network (AE) [65]. Its objective is to learn a non-linear mapping from a data space to a lower-dimensional feature space that could be used for clustering. First, DEC is pretrained with reconstruction loss as same as AE. Next, the encoder (or mapping function) without decoder is trained using Kullback–Leibler divergence minimization to improve the cluster results based on an auxiliary distribution computed during the pretrain stage. Recently, DeepCluster [62] is proposed for clustering without using the AE structure which iteratively clustered deep features and retrained the CNN to learn the pseudo-label by cluster assignments. Note that, these two methods are trained using clustering loss.

Another approach is based on the pairwise similarity and dissimilarity instead of clustering loss, where the similarity and dissimilarity are the distance between two samples of the same and different clusters, respectively. Siamese Network [63] and Deep Adaptive Clustering (DAC) [64] are trained to obtain the embedded features by iteratively cluster the samples by their pairwise

similarity and learning to minimize the similarity within-group and maximize the dissimilarity between samples of a different group. Although Siamese Network and DAC use a similar network structure, DAC has a constraint on selecting samples based on their cosine similarities while Siamese Network does not have any constraints. DAC could learn without explicit cluster labels while Siamese Network requires them to learn discriminative features.

An advantage of the deep neural network-based clustering methods is the representation ability from high dimensional feature space. Clustering results of many traditional clustering methods that are mainly based on similarity measures, for example, distance functions, are usually degraded when there is a high dimensional feature space. These deep neural network-based methods, however, are trained to extract low-dimensional features from high-dimensional feature space which is driven by data. In the case of the HMEs clustering problem, there is a challenge in the generalization solution. These deep neural network-based clustering methods are designed to learn category discriminative features for clustering images while HMEs have unlimited compositions of symbols and structured relations. Thus, HMEs clustering requires not only symbol/object discriminative features but also spatial information of MEs.

### **3.4.1 Multi-scale object localization and classification features**

In this section, we make an overview of multi-scale object localization and classification features using weakly supervised learning.

#### ***3.4.1.1 Multi-scale object localization and classification***

To deal with multi-scale object detection, one approach is to use a combination of multi-level features from different Convolutional Neural Networks (CNNs) [66]. The method, however, requires a large amount of training data in different scales since the networks learn features for each scale independently. Another approach is to use a single CNN to extract features from multiple scales samples of input image [67] [68] and aggregates them into a single feature. For this approach, the networks are designed to detect from a fixed scale object, and inference for multi-scale objects is obtained from the proper scale depended on input images.

Faster R-CNN [69] used ROI pooling to rescale the windows of objects into fixed-size windows for dealing with the problem of multi-scale object detection. The method also used a single CNN to learn, however, it needs location information to train object location prediction.

Fully Convolutional Networks [70] use multiple levels of CNN features to classify each pixel location in an input image. It could be used to detect multi-scale objects. The method, however, needs supervised labels for each pixel.

#### 3.4.1.2 Weakly supervised learning for object localization and classification

Weakly supervised learning [67] [71] is a learning method for CNN object localization and classification. The method applies a global max-pooling layer (GMP) [67] or global average pooling (GAP) [71] to aggregate local features by CNN through the spatial dimensions. The aggregated features represent object classes that exist or not in the images. Let  $F_c$  be the feature by CNN, GMP and GAP is represented as follows:

$$GMP(F_c) = \max_{x,y} F_c(x, y) \quad (18)$$

$$GAP(F_c) = \text{average}_{x,y} F_c(x, y) \quad (19)$$

where  $x, y$  represents the spatial location of the local features by CNN.

Without location information, weakly supervised learning uses the labels which indicate each class exists or not in the images. For each class  $c$ , two possibilities exist in the input image or not. Let  $t_c$  be the binary target of  $\{0, 1\}$ , where 0 indicates that the class  $c$  does not exist and 1 indicates that it does exist in the image,  $y_c$  be the classification probability of class  $c$ , the binary cross-entropy loss for the class  $c$  is obtained as follows:

$$BCE_c = -(t_c \log y_c + (1 - t_c) \log(1 - y_c)) \quad (20)$$

Binary cross-entropy loss  $BCE$  for all the classes calculated by (4) is used to optimize the weighting parameters for the networks.

$$BCE = \sum_c BCE_c \quad (21)$$

#### 3.4.1.3 Class activation map

The networks trained by weakly supervised learning can be used to generate class activation maps (CAMs) [71] (or class score maps [67]) by applying to remove the global pooling layer. The output is the activation maps for every class representing the prediction of the class in each location of the output map. CAMs retain both the classification of objects and the spatial structure of these objects.

#### 3.4.1.4 Spatial pyramid pooling

Spatial pyramid pooling (SPP) [72] divides an input image of arbitrary size into a multi-level of local spatial bins, concatenates the features from the spatial bins into a fixed-size feature

representation. The length of features by SPP is equal to  $number\_of\_bins \times feature\_depth$ . SPP maintains spatial information of the features by high-level spatial bins. If only the first level of the  $1 \times 1$  bin is used, spatial information is removed from the features.

### 3.4.2 Method

We propose a Hierarchical Spatial Classification (HSC) features for clustering HMEs. The CNN is trained by weakly supervised labels to extract CAMs that contain both the location and classification of symbols in HMEs. CAMs are aggregated by a multi-level of Spatial Pooling to obtain HSC features.

Note that, handwriting symbols in an HME may have different scales according to their roles. For example, the symbols in subscript or superscript are smaller than those in the baseline, which raises a challenge for CNN to detect characters at multiple scales. Thus, we design a single feature extractor for multi-scale inputs using a single CNN to extract features from multiple scales of an input HME image. Our proposed network could choose an appropriate scale for extracting useful features.

Moreover, as weakly supervised learning gathers the features from all the locations of an image, it should focus on learning from the locations that symbols exist. We use an attentive pooling layer that focuses on extracting all locations by each symbol category, which helps our network localize not only a single location but also all locations if a symbol category appears at multiple locations.

#### 3.4.2.1 Hierarchical Spatial Classification features for clustering

Inspired from SPP, we use the multi-levels of spatial pooling for extracting the Hierarchical Spatial Classification (HSC) feature from CAMs. These spatial poolings, however, allow various aspect ratios instead of  $1:1$  as in the original SPP. Figure 1 shows an example of two levels of spatial pooling  $\{2 \times 3, 3 \times 4\}$  for extracting HSC features from CAMs of the symbol ‘*i*’ in two different samples of the same formula. HSC features eliminate the variance of symbol location in different writing styles since it produces the same result if the location of the object is inside the spatial bin. In Figure 11, although the locations of the symbol ‘*i*’ are different for the two samples, the feature by  $2 \times 3$  spatial pooling of the two images are the same.

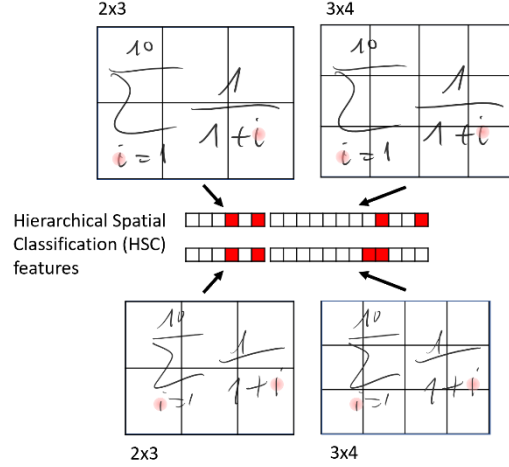


Figure 11. HSC features from Class Activation Map of class ‘i’.

The partition size (e.g.  $2 \times 3$ ,  $3 \times 4$ ) of spatial pooling affects clustering. Features by coarse partition robust with a variance of symbol locations in HME, but it also less spatial sensitivity to discriminate complex HMEs. The effect is reversed for the fine partition. Therefore, HSC features by multi-levels of spatial pooling from coarse to fine could compensate for the disadvantage of each single spatial pooling level. In Figure 11, although  $3 \times 4$  bins produce the different features of the symbol ‘i’ for the same formula,  $2 \times 3$  bins could produce the same features so that the hierarchical of  $\{2 \times 3, 3 \times 4\}$  bins could produce more robust features.

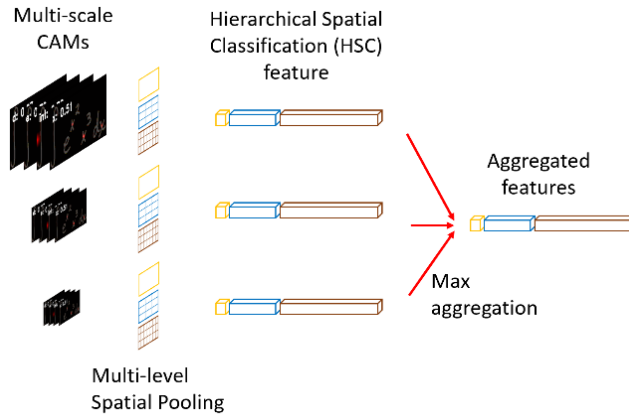


Figure 12. Multi-scale aggregated HSC features for clustering.

Multi-scale CAMs vary in their sizes since they are dependent on the input sizes of HMEs. Thus, we apply multi-level spatial pooling for each scale of CAMs feature to conform the multi-

scale CAMs into the HSC features of the same size and aggregate the features by max aggregation. Figure 12 illustrates the detailed approach.

#### 3.4.2.2 Overall of our proposed network

The overall architecture of the proposed network is shown in Figure 13. Input images are down-sampled by a half and one-fourth along each spatial dimension to obtain multi-scale inputs. Multi-scale CAMs are extracted from multi-scale inputs by a single CNN-based feature extractor that reduces the complexity of our network and learns the shared features from multi-scale inputs. Next, a global attentive pooling layer is employed to obtain the fixed-size feature vectors ( $k$ -dimension where  $k$  is the number of categories) from multi-scale CAMs. These fixed feature vectors are aggregated by the max aggregation to retain the best-detected symbol by multi-scale inputs. We use weakly supervised learning to train our network without using symbol location information, which removes human labeling costs for preparing supervised locations.

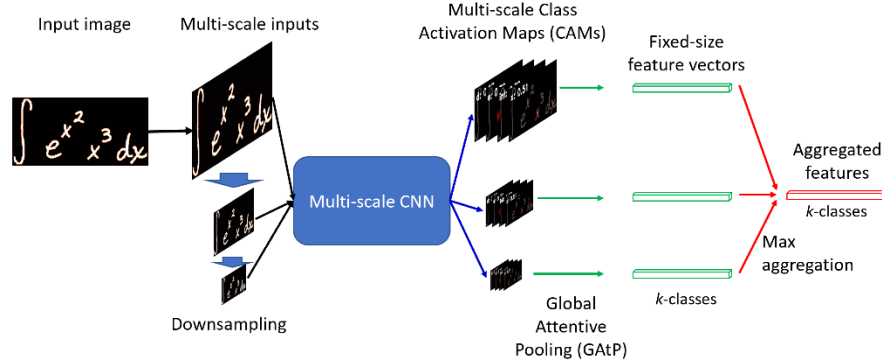


Figure 13. Overall of the proposed network

#### 3.4.2.3 Multi-scale CNN for extracting CAMs

Multi-scale CAMs represent both the location and classification of symbols in an HME image, as shown in Figure 14. The network extracts these features for object localization and classification presented in 3.4.1.1. In an HME, the dependencies between symbols are also essential to reduce ambiguity for classifying the symbols. For example, recognizing bracket symbols or fraction symbols requires not only local information of the symbols but the context of other adjacent symbols. Therefore, we concatenate the last two levels of convolutional layer outputs to use the local context of symbols for recognition. The symbol level block as shown in Figure 14 consist of convolution and pooling layers with the output is the symbol-level feature by concatenating the output of the last two convolution layers. The symbol-level features are then

classified by the expression-level block. The detailed architectures of the blocks are shown in Table 4. The symbol-level block is configured to handle the input symbols in  $32 \times 32$  to recognize small symbols in an HME. The expression-level block outputs the probability for the symbol characters. The CNN is learned by weakly supervised learning, where a global pooling layer (GAP, GMP) is added after the classifier.

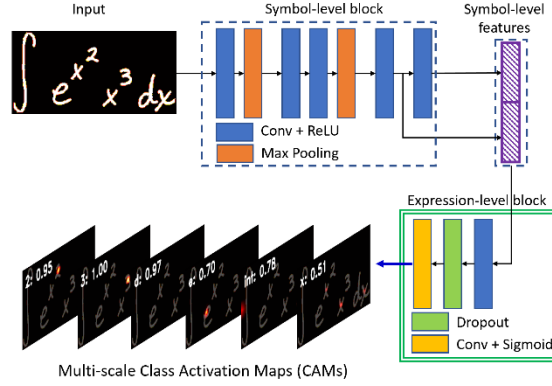


Figure 14. Multi-scale CNN for extracting CAMs

Table 4. Network configuration

Module	Type	Configurations
Symbol-level block	Convolution - ReLU	#maps:64, k:3×3, s:1, p:1
	MaxPooling	#k:2×2, s:2
	Convolution - ReLU	#maps:128, k:3×3, s:1, p:1
	Convolution - ReLU	#maps:256, k:3×3, s:1, p:1
	MaxPooling	#k:2×2, s:2
	Convolution - ReLU	#maps:512, k:8×8, s:1, p:1
	Convolution - ReLU	#maps:512, k:3×3, s:1, p:1
Expression-level block	Convolution - ReLU	#maps:1024, k:1×1, s:1, p:1
	Convolution - Sigmoid	#maps:101, k:1×1, s:1, p:1

#### 3.4.2.4 *Global attentive pooling*

The networks with GMP, however, only learn from the highest activation location for each class. On the other hand, the networks with GAP learns from all the locations in the CNN feature. Restricting GAP in the object region of interest (ROI) helps the model learns well [71], but it requires location information for all the objects.

To enhance localization and classification, we extract the attention of input images to train the classifier through a global attentive pooling layer (GAtP). The global attentive pooling layer aggregates the features for learning each separate class. For each class, an attentive map focuses



on the locations of the symbols in the class, aggregates the local features in these locations into a single feature. The attentive pooling makes the networks learn from multiple locations where the symbols occur. Therefore, it can overcome the problem of GAP and GMP.

First, we obtain the CAMs denoted as  $M_c$  for all the classes by applying the classifier to the CNN features. For a class  $c$ , we multiply  $M_c$  with the CNN features  $F$  to extract attentive features for class  $c$ . We aggregate the features by an average pooling on the attentive features as follows:

$$Att_{F_c} = \frac{\sum_{x,y} M_c(x,y)F(x,y)}{xy} \quad (22)$$

where  $x, y$  denotes the spatial dimensions of CNN features. The aggregated feature is fed to the classifier for training the class  $c$ .

The output  $y_c$  of the class  $c$  by the classifier  $f$  with attentive features  $Att_{F_c}$  is obtained as follows:

$$y_c = \langle f(Att_{F_c}), u_c \rangle \quad (23)$$

where  $u_c$  is the one-hot vector of class  $c$  in total  $k$  classes,  $\langle a, b \rangle$  is the inner product of vector  $a$  and vector  $b$ . The network is trained by applying the  $BCE_c$  loss (1) using  $y_c$  obtained by (4).

For implementation, let the attentive features for all the classes  $Att_F = [Att_{F_1} \quad Att_{F_2} \quad Att_{F_k}]$ , the output  $y = [y_1 \quad y_2 \quad y_k]$  is obtained as follows:

$$\begin{aligned} y &= [\langle f(Att_{F_1}), u_1 \rangle \quad \langle f(Att_{F_2}), u_2 \rangle \quad \langle f(Att_{F_k}), u_k \rangle] \\ &= \text{diag}([f(Att_{F_1}) \quad f(Att_{F_2}) \quad f(Att_{F_k})]) \\ &= \text{diag}(f(Att_F)) \end{aligned} \quad (24)$$

where  $\text{diag}(A)$  is the operator to get elements on the main diagonal of matrix  $A$ .

Figure 15 shows the details of our global attentive pooling layer. From top to bottom, the multi-scale HSC features of each class are multiplied with symbol-level features to obtain an attentive feature vector by (22). Next,  $k$  attentive feature vectors are fed into the expression-level block to obtain  $k$  classified vectors as presented by (23) which are combined to form a fixed-size feature vector by (24).

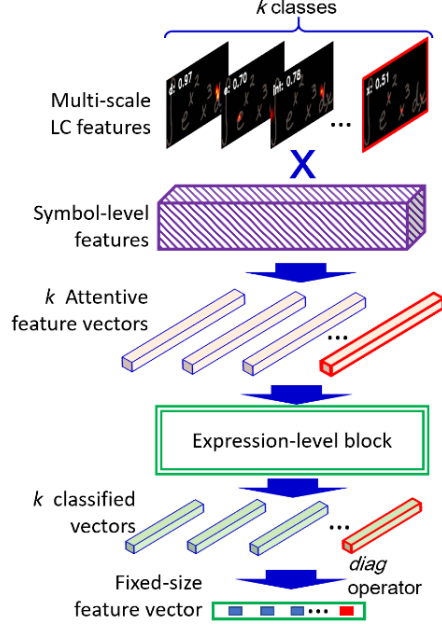


Figure 15. Global attentive pooling (GAtP)

#### 3.4.2.5 *Combination of GAtP and GMP*

Global attentive pooling may not be trainable at the early epochs of training since the attentive maps produce low activations. Global attentive pooling also produces low activations for the classes which do not exist in an image. Hence, negative learning is not well performed. We overcome the problem by a combination of the output activations by both GAtP and GMP as follows:

$$y_c = \frac{1}{2} (\langle f(Att_{F_c}), u_c \rangle + \langle f(GMP(F)), u_c \rangle) \quad (25)$$

Combining output activations makes the networks learn from both the loss provided by GAtP features and that by GMP features. At early epochs, networks benefit from the GMP loss and at the final states, networks benefit from the GAtP loss.

### 3.5 Marking Cost Function

In this study, we propose a marking cost function for evaluating clustering-based marking assistance without requiring data from human participants.

Assume that  $W$  is a set of clusters  $\{w_1, w_2, \dots, w_K\}$ , with  $K$  being the number of clusters and  $C$  is a set of classes  $\{c_1, c_2, \dots, c_J\}$ , with  $J$  being the number of classes. For each cluster  $w_k$  in  $W$ , let  $M_k$  be the set of samples belonging to the major class  $c_j$  (the class  $c_j$  has the most samples) in the cluster  $w_k$  as shown in (26):

$$|M_k| = \max_{1 \leq j \leq J} \{|w_k \cap c_j|\} \quad (26)$$

Then, the purity is the accuracy of this assignment on the whole dataset divided by the total number of samples  $N$ , as shown in (27):

$$\text{Purity}(W, C) = \frac{1}{N} \sum_{k=1}^K |M_k| \quad (27)$$

High purity is easy to achieve when the number of clusters is huge. For example, if  $K$  equals  $N$ , we obtain the purity of one. Thus, the purity itself does not show the clustering quality.

We propose a cost function to evaluate the contribution of a marker during the marking process. Our cost function addresses the deficiency of the purity score by considering both the purity and the number of clusters. Given a cluster of answers, if there exist more than two different sets of answers, the marker tends to weed out the minor answers and retain the major one. Thus, the major answers just need a single marking operation while the minor ones require the same amount of effort as the manual approach.

Marking may depend on the size (or length) of answers and the nature of questions. However, we neglect these issues as the first step by considering a simple scenario of verifying and marking answers, where the task of verifying is to find different answers inside a cluster and the task of marking is to compare an answer with the correct answer. Both the tasks need to detect differences between an answer with other(s). Though detecting differences may depend on the sizes of expressions, we can take their average and assume little dependence on the question except for specific cases.

Assuming that a verification cost and a marking cost (in terms of time) are incurred per answer. Let  $T$  be the average marking cost of all answers. The verification cost is expected to be lower than the marking cost. We assume that there is exist a real number  $\alpha$  ( $0 < \alpha \leq 1$ ) so that the average verification cost is  $\alpha T$ . Considering a cluster  $w_i$ , which is separated into a set of major answers  $M_i$  and some sets of minor answers, its marking cost is calculated as shown in (28):

$$\text{cost}(w_i, C) = C_{\text{marking}} + C_{\text{verification}} = (1 + |w_i| - |M_i|) \times T + \alpha |w_i| \times T \quad (28)$$

where  $C_{\text{marking}} = (1 + |w_i| - |M_i|) \times T$  is the cost of marking the single major set and  $|w_i| - |M_i|$  samples in the minor sets, and  $C_{\text{verification}} = \alpha |w_i| \times T$  is the cost of verifying all the answers in the cluster  $w_i$ .

Thus, the cost of the whole dataset is given in (29):

$$\text{cost}(W, C) = \sum_{w_i \in W} \text{cost}(w_i, C) \quad (29)$$

$$= (K + N - \sum_i |M_i|) \times T + \alpha NT$$

We rewrite (29) using the purity term as shown in (30):

$$\begin{aligned} \text{cost}(W, C) &= \sum_{w_i \in W} \text{cost}(w_i, C) \\ &= (K + (1 - \text{Purity}(W, C)) \times N) \times T + \alpha NT \end{aligned} \quad (30)$$

To estimate the worst case, we assume that the verification cost is as large as the marking cost ( $\alpha = 1$ ) and rewrite the cost function in (30) as shown in (31):

$$\text{cost}(W, C) = (K + (1 - \text{Purity}(W, C)) \times N) \times T + NT \quad (31)$$

Consider the following cost:

$$\text{SimplifiedCost}(W, C) = (|W| + (1 - \text{Purity}(W, C)) \times N) \times T \quad (32)$$

**Proposition:**  $|C|T \leq \text{SimplifiedCost}(W, C) \leq NT$

**Proof:**

From the (27) and (31) we obtain (33):

$$\text{SimplifiedCost}(W, C) = (|W| + N - \sum_{k=1}^K |M_k|) \times T \quad (33)$$

Firstly, we prove that the minimum value of the cost function is  $|C|T$ . Since the purity cannot exceed 1, we get:

$$\Rightarrow N - \sum_{k=1}^K |M_k| \geq 0 \quad (34)$$

We consider two cases:

1) If  $|W| \geq |C|$ , we add  $|W|$  to both sides of (35):

$$\Rightarrow |W| + N - \sum_{k=1}^K |M_k| \geq |W| \geq |C| \quad (35)$$

2) If  $|W| < |C|$ , there exists  $|C| - |W|$  classes that are not assigned to any cluster. Hence, there are at least  $|C| - |W|$  samples that belong to minor sets. The number of samples which belong to minor sets is calculated as  $N - \sum_{k=1}^K |M_k|$ . Therefore, we obtain (36):

$$\begin{aligned}
N - \sum_{k=1}^K |M_k| &\geq |C| - |W| \\
\Rightarrow |W| + N - \sum_{k=1}^K |M_k| &\geq |C|
\end{aligned} \tag{36}$$

The equation is obtained when the clustering algorithm produces a perfect clustering result (purity is 1) and the number of clusters is exactly equal to the number of classes.

Secondly, we prove the maximum value of the cost function is  $NT$ . In each cluster, the number of samples in the major set is greater than or equal to 1. Hence, the total number of samples that belong to the major set is greater than or equal to the number of clusters. We obtain (37):

$$\sum_{k=1}^K |M_k| \geq |W| \Rightarrow |W| - \sum_{k=1}^K |M_k| \leq 0 \tag{37}$$

Adding  $N$  to both sides of Eq. (37), we obtain Eq. (38):

$$|W| + N - \sum_{k=1}^K |M_k| \leq N \tag{38}$$

The equation is obtained when the number of clusters equals the number of samples ( $|W| = N$ , i.e., each sample belongs to its own cluster).

Since the maximum value of  $\text{cost}(W, C)$  in (31) is  $2NT$ , we normalize the value range of Eq. (17) into the interval of  $[0,1]$  by dividing it by  $2NT$ . Our final cost function is shown in (39):

$$f(W, C) = \frac{K}{2N} + \left(1 - \frac{1}{2} \text{Purity}(W, C)\right) \tag{39}$$

Note that  $f$  approaches 1 in the worst case, implying that the marking cost approaches the manual one-by-one marking cost. In the HME clustering problem, we try to minimize this cost function to the number of clusters  $K$  and purity. Note that this cost function indicates the worst case since we assumed  $\alpha = 1$  and  $\text{cost}(W, C) = 2NT$  as shown in (31). By designing an effective user interface for makers, the value of  $\alpha$  can be decreased and the marking cost is reduced.

### 3.6 Incremental refining clustering

In practice, it usually needs a marker to verify and regrade “impure” answers within a cluster during the marking process where “impure” answers are the answers which are not assigned to

the most frequent class in that cluster. If an “impure” answer is detected, it is reallocated to a separated set by the examiner. This interaction must be reflected the further clustering. For the remaining unmarked clusters, an element is moved to one of the newly created sets if its distance to the centroid of that new set is shorter than to the centroid of that cluster. The algorithm is presented in Algorithm 1.

**Algorithm 1**

Input: Set of pair of clusters and centroids  $W, U = \{(w_1, u_1)(w_2, u_2), \dots, (w_k, u_k)\}$ ,

1. Initialize the set of “impure” subsets of samples  $S = \{\emptyset\}$ .
2. Choose an initial unmarked cluster  $w_i$  for the marker.
3. Get the marking results for that cluster in which  $M_i$  is the set of major answers and  $m_i = \{m_{i1}, m_{i2}, \dots, m_{ir}\}$  is the set of minor answers.
4. Append  $m_i = \{m_{i1}, m_{i2}, \dots, m_{ir}\}$  into  $S$
5. For each unmarked clusters  $\langle w_k, u_k \rangle \in W, U$ 
  - a. For each data point in that cluster  $p \in w_k$
  - b. If there exist  $i1 \leq id \leq ir$  such that  $\text{distance}_{\text{cluster}}(p, m_{id}) < \text{distance}_{\text{set}}(p, u_i)$  then append  

$$S = \{\dots, m_{id} \cup \{p\}, \dots\}$$
6. Repeat step 2 until all samples have been marked

Figure 16 illustrates the algorithm. Here, we define the distance between a point and a cluster to be the distance from the point to the centroid of that cluster. The distance from a point to a set of points is the minimum distance from that point to each point in the set of points.

We use the within-cluster mean of squared distance to choose the cluster for marking. For each pattern  $p_j$  in a cluster  $w_i$  with its centroid  $\mu_i$ , the within-cluster mean of squared distance is calculated as in (40).

$$\text{WCMSD}(w_i) = \frac{1}{|w_i|} \sum_{l=1}^{|w_i|} \|p_l - \mu_i\|^2 \quad (40)$$

Then, we choose the initial cluster whose WCMSD is the smallest.

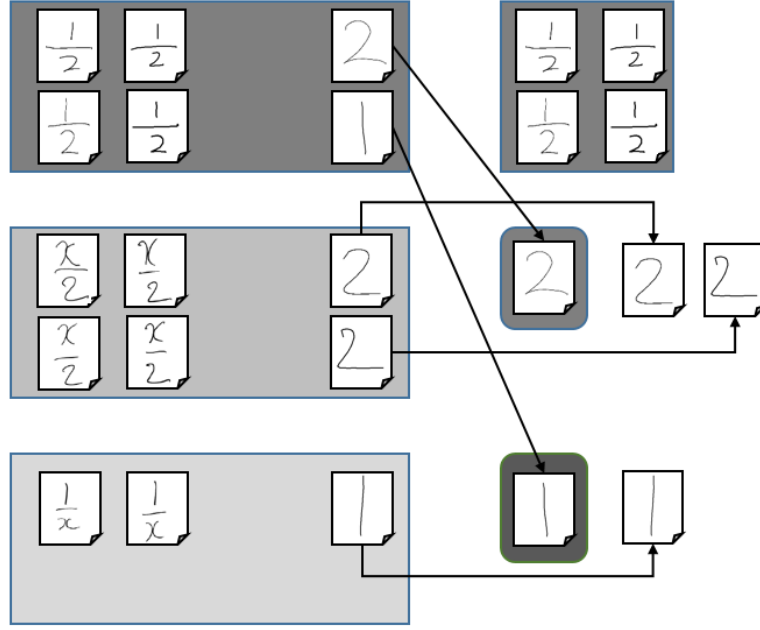


Figure 16. Incremental refinement approach

### 3.7 Datasets

We use three datasets of HMEs to evaluate the features and methods. These datasets share the same 101 Math symbol categories mentioned in [6]. The first dataset named “Dset\_22Qs” was collected from 23 students (19 males and 4 females). Each student wrote three prepared answers: a correct answer and two incorrect (partially incorrect and totally incorrect) answers for each of the 22 questions, resulting in a total number of samples:  $23 \times 3 \times 22 = 1,518$ .

The second dataset named “Dset\_50” was collected from 21 students to evaluate the performance with a larger number of different answers. This data set was originally collected to test HME recognition on three different user interfaces: (1) without any guiding line; (2) with a center line; and (3) with center, top, and bottom lines. Each student wrote 50 different HMEs on the three interfaces, thereby generating  $21 \times 50 \times 3 = 3,150$  HME patterns. Although its original purpose was different, Dset\_50 can be used for evaluating the clustering of HMEs. The details of these datasets are shown in Table 5.

The third dataset was sampled from the CROHME 2016 dataset [8] but it was not enough so it was augmented by synthetic patterns from LaTeX sequences and isolated handwritten symbol patterns in CROHME 2016. This augmented data set is named “Dset\_Mix”, and is organized so that real (handwritten) HME patterns and synthetic patterns are mixed in a way that each question has a few correct answers and several incorrect answers. Although real answers are the best,

collecting correct and incorrect answers requires data from real examinations with participants' agreements. Therefore, we took the synthetic approach to augment real HME patterns like [33] and made the dataset available for public use. We prepared 200 answers for each of 10 presumable questions, which included handwritten and synthetic answers, as well as correct and incorrect answers. This sample size is set according to the number of students in each grade in a typical school. We published the dataset along with the list of the 10 presumable questions<sup>1</sup>.

Table 5. Specifications of Dset\_22Qs and Dset\_50

<b>Dataset</b> <b>Statistical item</b>	<b>Dset_22Qs</b>	<b>Dset_50</b>
<b># students</b>	23	21
<b># questions</b>	22	1
<b># answers per question</b>	3	150
<b>total</b>	1,518	3,150

Figure 17 shows the process of creating a synthetic pattern using the tool described in [73]. Given a LaTeX sequence, the tool first prepares a template by reflecting the sizes and positions of the symbols based on their relations to each other (horizontal, superscript, subscript, above, below, inside). Second, it translates bounding boxes in the template randomly. Third, it resizes and fills handwritten patterns into the prepared template. Finally, symbol patterns from CROHME 2016 are filled into the template.

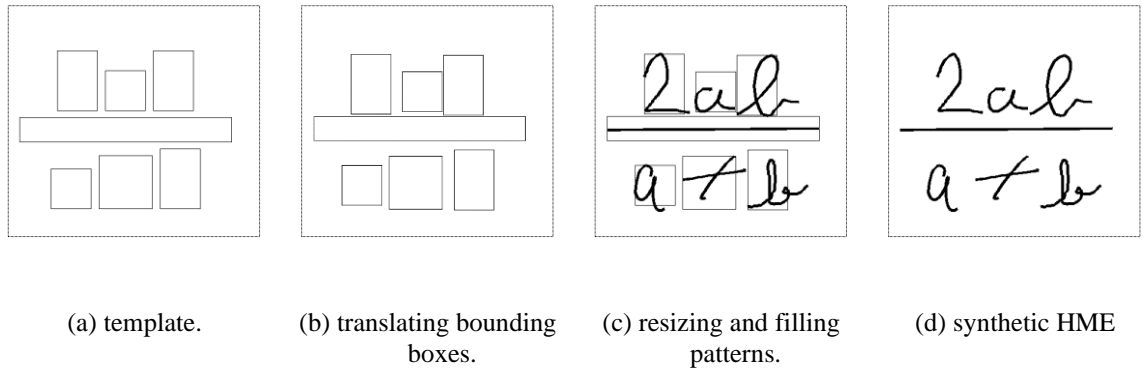


Figure 17. The process to synthesize an HME

<sup>1</sup> [http://tc11.cvc.uab.es/datasets/Dset\\_Mix\\_1](http://tc11.cvc.uab.es/datasets/Dset_Mix_1)



Dset\_Mix is divided into ten subgroups corresponding to ten presumable questions. Each subgroup stores one or two correct answers and one to five incorrect ones with each having one or multiple HME patterns. Table 6 shows the number of HME categories and patterns in each subgroup. Figure 18 shows some sample patterns in subgroup 1 for the presumable question 1 (PQ 1). According to Table 3, Dset\_Mix consists of 2,000 HMEs (263 handwritten HMEs from CROHME 2016 and 1,737 synthetic patterns).

correct answer  
(handwritten pattern)

incorrect answer  
(handwritten)

incorrect answer  
(handwritten)

incorrect answer  
(synthetic pattern)

incorrect answer  
(synthetic)

incorrect answer  
(synthetic)

Figure 18. Samples in Dset\_Mix. (PQ 1: If a quadratic function  $ax^2 + bx + c = 0$  has two roots, what is the general form of the roots?)

Table 6. Dset\_Mix dataset

Subgroup No.	1	2	3	4	5	6	7	8	9	10
Statistical item										
# categories of correct answers	2	1	1	1	1	1	2	1	1	1
# handwritten patterns of correct answers	26	0	0	20	20	20	20	27	0	0
# synthetic patterns of correct answers	21	40	50	20	35	10	81	49	50	50
# categories of incorrect answers	8	4	5	4	4	6	2	5	3	3
# handwritten patterns of incorrect answers	21	18	3	19	2	39	1	27	0	0
# synthetic patterns of incorrect answers	132	142	147	141	143	131	98	97	150	150
# total answers	200	200	200	200	200	200	200	200	200	200

### 3.8 Experiments and Results

In this section, we seek to find the best feature type or feature combination and the best combination method.

#### 3.8.1 Experiments on individual feature types

The number of symbol categories in the datasets is 101 so that the dimension of the bag-of-symbols feature vector is 101 ( $S = 101$ ). We trained a CNN on the isolated symbols in CROHME 2013 and achieved a symbol recognition rate of 89.39% on CROHME 2014.

The clustering performance of the feature types was evaluated individually. For Dset\_22Qs, we applied the clustering method question by question (the number of classes is 3 for each question). For Dset\_50, we applied the clustering method on the whole dataset so that the number of classes was 50. For Dset\_Mix, we employed the clustering method question by question (the number of classes varied from 4 to 10). We used  $k$ -means++. For each type, we fixed  $k$  (the only parameter of  $k$ -means++) to be equal to the number of classes for evaluating its discrimination ability, although the number of true classes is unknown for real situations. The Euclidean distance was used for clustering.

Table 7 shows the purity by the individual feature types for Dset\_22Qs, Dset\_50, and Det\_Mix in terms of the average purity for the 22 questions in Dset\_22Qs, the purity for 50 ME categories in Dset50, and the average purity for ten sets in Dset\_Mix. The results show that the bag-of-symbols feature achieves the highest purity among the individual feature types on the three datasets. Dset\_Mix shows similar purity levels as Dset\_22Qs and Dset\_50 for all the feature types, which means that it may be used for further evaluation.

Positional BoS, Positional BoR, and their combination are inferior to simple Bag-of-symbols, except their combination for Dset\_Mix. The reason seems that concatenating the bag-of-features of all the bins makes their dimension huge and the features sparse as known as the curse of dimensionality. Hence, the clustering performance is not improved.

Table 7. Purities of individual features (mean  $\pm$  standard deviation).

Feature type \ Dataset	Dset_22Qs	Dset_50	Dset_Mix
Directional feature (Dir)	0.6465 $\pm$ 0.1835	0.7803	0.7046 $\pm$ 0.1690
<b>Bag-of-symbols (BoS)</b>	<b>0.7642 <math>\pm</math> 0.1779</b>	<b>0.8587</b>	<b>0.7175 <math>\pm</math> 0.1111</b>
Bag-of-relations (BoR)	0.5743 $\pm$ 0.1171	0.3479	0.4650 $\pm$ 0.0705
Bag-of-positions (BoP)	0.5572 $\pm$ 0.0098	0.2010	0.5415 $\pm$ 0.0569
Positional BoS	0.7213 $\pm$ 0.1311	0.8181	0.7075 $\pm$ 0.0842
Positional BoR	0.5114 $\pm$ 0.1475	0.5532	0.5430 $\pm$ 0.0633
Positional BoS + BoR	0.7111 $\pm$ 0.1854	0.7532	0.7295 $\pm$ 0.1587

### 3.8.2 Experiments on feature combination

We evaluated the methods of combining individual feature types. To evaluate Weighted\_Comb, we needed to train its weights, although we did not need to train Normalized\_Comb and Standarize\_Comb. Therefore, we applied the 5-fold cross-validation for Weighted\_Comb. For the cross-validation, one subset was reserved for testing, and the remaining four subsets were used for determining the optimal set of weights by employing the grid search technique and measuring the performance on the testing subset in each round. The average across

all the five testing subsets was taken. The entire Dset\_50 was used for training to obtain the optimal weights, which were applied for Weighted\_Comb on Dset\_22Qs and Dset\_Mix. For Normalized\_Comb and Standarize\_Comb, we employed these methods directly.

There can be two different ways to form the five folds. One way is to split the dataset by the HME categories, and the other is to split it by the writers. The former evaluates the clustering performance on “unseen” HME categories, whereas the latter evaluates the clustering performance on “unseen” writing styles. In this study, we take the former approach and split the dataset into five subsets by the HME categories as proper clustering-based marking assistance should be able to cluster unseen HME categories.

Table 8 shows the mean purity and standard deviation for combining various feature types by each combination method. As the bag-of-symbols is the best single feature type, we examined its combinations with others. We show the results in detail for Weighted\_Comb, but the other results are similar so that they are partially omitted. When this feature is combined with another feature type, the best results are achieved with the directional feature. When combined with two other feature types, the effect is small or even negative. When combined with the three other feature types, the effect is maximized. In this case, the weighting parameters of BoS, Dir, BoR, BoP are 0.45, 0.35, 0.15, 0.05, respectively. As we can see, BoS and Dir whose clustering performance is high are assigned with higher weights while BoR and BoP are assigned with lower weights.

As Weighted\_Comb learns the significance of each feature type through training samples, it performs better than Normalized\_Comb and Standardized\_Comb which normalizes or equalizes the role of each feature type. We also considered applying normalization and standardization before employing Weighted\_Comb after the feature normalization. The performance is significantly reduced with the purities are around 0.7485 (normalization) and 0.7400 (standardization). The reason is applying normalization and standardization methods will scale down the magnitudes of the features and make the distances in the feature space of BoS and Dir similar to BoR and BoP’s. Hence, the grid-search method seems difficult to find the most significant features and the performance is reduced. For example, after applying the standardization, the grid search method yielded the weight  $(w_{Dir}, w_{BoS}, w_{BoR}, w_{BoP}) = (0.2, 0.1, 0.5, 0.2)$  which the BoR was assigned the largest weight though the purity of this feature alone was not as good as BoS.

Table 8. Purities of individual and combined features (mean  $\pm$  standard deviation).

Feature types & Datasets Combination method	Feature types				Datasets		
	BoS	Dir	BoR	BoP	Dset_50 (5-fold cross-validation)	Dset_22Qs	Dset_Mix
Normalized_Comb	✓				0.8518 $\pm$ 0.1174	0.7182 $\pm$ 0.1032	0.7405 $\pm$ 0.1838
	✓	✓			0.8651 $\pm$ 0.1120	0.6856 $\pm$ 0.1058	0.7555 $\pm$ 0.1705
	✓	✓	✓		0.8635 $\pm$ 0.1138	0.6954 $\pm$ 0.1049	0.7580 $\pm$ 0.1608
	✓	✓	✓	✓	0.8786 $\pm$ 0.1149	0.7186 $\pm$ 0.1040	0.7835 $\pm$ 0.1645
Standardized_Comb	✓				0.8601 $\pm$ 0.1281	0.7540 $\pm$ 0.1039	0.7410 $\pm$ 0.1637
	✓	✓			0.8610 $\pm$ 0.1215	0.7557 $\pm$ 0.1033	0.7580 $\pm$ 0.1803
	✓	✓	✓		0.8705 $\pm$ 0.1250	0.7650 $\pm$ 0.1141	0.7585 $\pm$ 0.1692
	✓	✓	✓	✓	0.8737 $\pm$ 0.1179	0.7775 $\pm$ 0.1170	0.7530 $\pm$ 0.1735
Weighted_Comb	✓				0.8514 $\pm$ 0.1263	0.7642 $\pm$ 0.1779	0.7864 $\pm$ 0.0746
	✓	✓			0.9270 $\pm$ 0.1240	0.7868 $\pm$ 0.1121	0.8285 $\pm$ 0.1648
	✓		✓		0.8614 $\pm$ 0.1786	0.7745 $\pm$ 0.1167	0.7570 $\pm$ 0.0846
	✓			✓	0.8771 $\pm$ 0.1262	0.7751 $\pm$ 0.1105	0.7870 $\pm$ 0.1113
	✓	✓	✓		0.9167 $\pm$ 0.1163	0.7835 $\pm$ 0.1161	0.8045 $\pm$ 0.1667
	✓		✓	✓	0.8700 $\pm$ 0.1287	0.7768 $\pm$ 0.1176	0.7879 $\pm$ 0.1134
	✓	✓		✓	0.9206 $\pm$ 0.1182	0.7774 $\pm$ 0.1203	0.8175 $\pm$ 0.1576
	✓	✓	✓	✓	<b>0.9303 <math>\pm</math> 0.1189</b>	<b>0.7943 <math>\pm</math> 0.1061</b>	<b>0.8410 <math>\pm</math> 0.1515</b>
Normalized + Weighted_Comb	✓	✓	✓	✓	0.8113 $\pm$ 0.1839	0.7081 $\pm$ 0.1677	0.7485 $\pm$ 0.1658
Standardized + Weighted_Comb	✓	✓	✓	✓	0.8258 $\pm$ 0.1516	0.7468 $\pm$ 0.1774	0.7400 $\pm$ 0.1876

### 3.8.3 Experiments on marking cost for Dset\_Mix

To investigate the marking cost, we evaluated it on Dset\_Mix as Dset\_Mix has 200 answers for 10 presumable questions each, including correct and incorrect answers. It is expected to simulate the marking of handwritten math answers in schools. We applied the  $k$ -means clustering using the best feature combination method, i.e., Weighted\_Comb, for question by question. Although the true number of classes among the answers for each question is from 4 to 10 from Table 6, we cannot know this when marking the answers. Therefore, we considered the purity and the marking cost according to the parameter  $k$  of the  $k$ -means method. Table 8 shows the average purity and the marking cost of the 10 presumable questions along the logarithmic scale for the parameter  $k$ .

When  $k$  increases up to 12, the purity increases, and the marking cost decreases. In this case, an appropriate number of clusters helps the  $k$ -means algorithm adapt to the data distribution, and the marking cost decreases. When the number of clusters is 12, the marking cost achieves the minimum value (0.62). When the number of clusters further goes up over 12, the purity also continues to increase and approaches 1 but the marking cost starts to increase and approaches 1 eventually. As we expected, too many clusters make the  $k$ -means algorithm produce more groups of answers to be marked so that the marking cost increases. This result shows that the marking cost is determined as a trade-off between the clustering quality and the number of clusters.

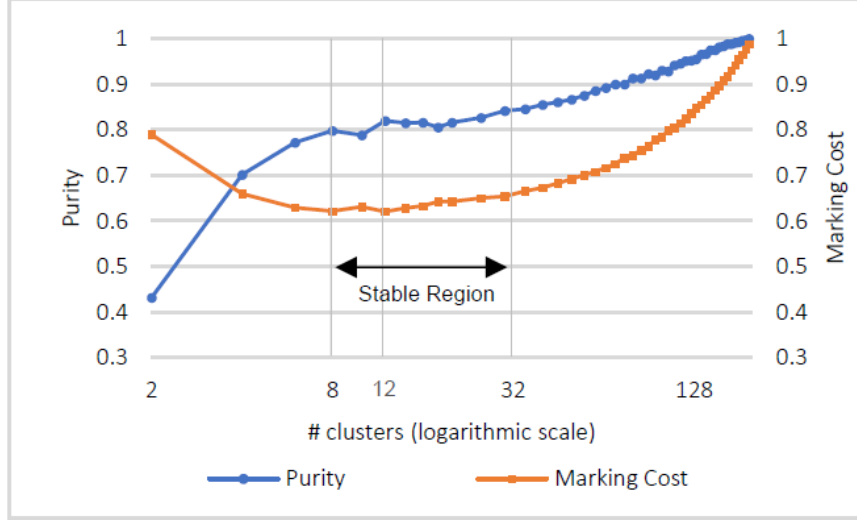


Figure 19. Purity and marking cost on Dset\_Mix.

The result also shows that the marking cost is rather stable between  $k = 8$  and  $32$  so that we can set the number of  $k$  as such, provided that the true number of classes are from 4 to 10 for 200 answers. Here, the “Stable Region” of a parameter  $p$  with an objective function  $f$  is the interval of the parameter in which the difference of the maximum and minimum values of the objective function  $f$  does not exceed a given threshold  $\eta > 0$ .

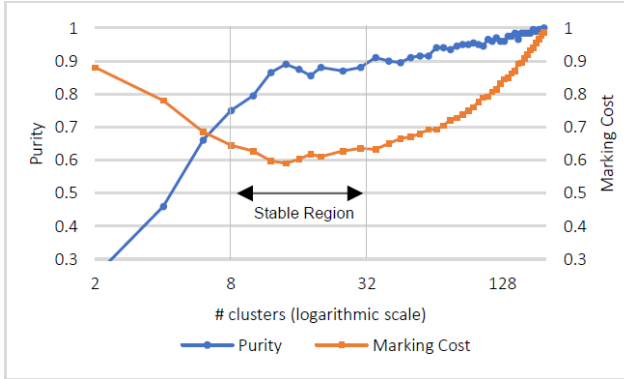


Figure 20. Purity and marking cost on PQ 1: If a quadratic function  $ax^2 + bx + c = 0$  has two roots, what is the general form of the roots? (the true number of answer classes is 4)

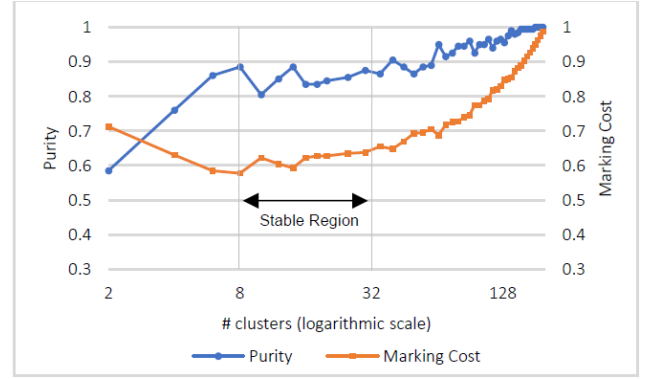


Figure 21. Purity and marking cost on PQ 7: Consider the circle, whose center is at  $(0,0)$  and radius equals  $r$ . Let  $(x,y)$  be a point on the circle such as the  $OP$  makes an angle  $\theta$  with the  $Ox$  axis. Represent  $x$  in terms of  $r$  and  $\theta$ . (the true number of answer classes is 4)

Figure 20 and Figure 21 show the cases for the presumable question (PQ), where the true number of answer classes is the largest (10), and PQ 7, where that is the smallest (4). For both

these cases, the marking cost is also stable between  $k = 8$  and 32 since the marking cost does not vary more than 0.1 ( $\epsilon = 0.1$ ) in this interval and it is reduced to about 0.6. This implies that the value of  $k$  is not so sensitive to the results for all the questions. Figure 22 visualizes the clustering result of the answers for PQ 1 by applying t-SNE.

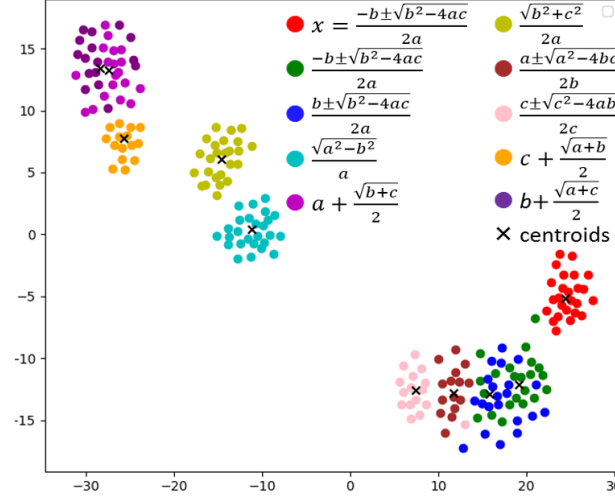


Figure 22. Visualization of clustering on question 1 (10 classes, 10 clusters)

The experiments show that the best combination of all the feature types can reduce the marking cost to about 0.6 by setting the number of answer clusters appropriately compared with the manual one-by-one marking.

### 3.8.4 Experiment on CNN based Spatial Classification Features

Since the deep learning-based models usually require a huge amount of data, we must utilize a dataset we use CROHME 2016 and CROHME 2019 for evaluating the CNN-based Spatial Classification Features. In CROHME 2016, we reserve the formulas from the folder “expressmatch” of the training set which contains the most samples per formula for clustering evaluation. In CROHME 2019, we reserve 49 most frequent formulae as a total of 1264 samples for clustering evaluation. All the samples are considered as the answers to a question. The remaining samples from the training set are used for training. The validation set is the same as the original configuration. The formula of test samples is not available in train samples.

We also do the experiments on Dset\_22Qs and Dset\_50 to compare with the previous features. We consider all the samples belong to the same question. For the two databases, we use 5-fold cross-validation divided by writers for training, validation, and testing, where the ratio of train:validation:test sets is 3:1:1.

All the input images are scaled into 128-pixel height with maintaining aspect ratio. There are images whose width larger than 800 pixels after scaling, they are rescaled to 800-pixel width. Multi-scale images are created by applying average pooling to the input images. There are three input images of the 128-pixel height, 64-pixel height, and 32-pixel height. The number of output classes for all three databases is 101 classes as the number of symbols in the CROHME dataset.

For spatial pooling, we extract a multi-level pooling size of  $(1 \times 1)$ ,  $(3 \times 5)$ ,  $(3 \times 7)$ , and  $(5 \times 7)$  feature map bins, where  $(1 \times 1)$  spatial pooling does not retain spatial information, the other spatial pooling could retain the spatial information.

We apply cosine distance to measure the dissimilarity between the HSC features extracted from two HME images. We adopt k-means++ as a clustering method to clustering the HME images. To cluster the HME images by k-means++, for each HME image, we calculate the cosine distances of it to all other HME images and use the distances as the representation of the HME images.

#### 3.8.4.1 *Clustering experiments*

To evaluate the feature fairly, we set the number of clusters for hierarchical clustering by the same as the number of formulas in the testing sets. For each question in Dset\_22Qs, we cluster the answers into three groups and report the average purity overall the questions. For Dset\_50, we cluster the answers into 50 groups. For CROHME 2016 and CROHME 2019, we cluster them into 36 groups and 49 groups, respectively.

The clustering result is shown in Table 9. The method using HSC features of CNN outperforms the online Bag of Features (Online BoFs) and the offline Bag of Features (Offline BoFs) by the large margins on both the Dset\_22Qs and Dset\_50. For CROHME 2016 and CROHME 2019, our method also yields the purity of 0.99 and 0.96, respectively. The result of the CROHME dataset shows the robustness and high generalizability of the learned features over handcrafted features.

Table 9. Clustering results of HSC feature compared with other methods (purity)

Method	Dset_22Qs	Dset_50	CROHME 2016	CROHME 2019
Online BoFs	-	0.92	-	-
Offline BoFs	0.76	0.87	-	-
k-means+PCA	0.51	0.45	0.48	0.47
DAC	0.64	0.61	0.77	0.76
Siamese	0.65	0.79	0.88	0.76
HSC feature	<b>0.88</b>	<b>0.98</b>	<b>0.99</b>	<b>0.96</b>

We also reproduced the experiments with other image clustering methods as in Table 9. The first method applies PCA with the number of components set to the number of clusters to reduce the dimensionality of the images. Then the reduced features are clustered by k-means++. The second method is DAC, which learns to cluster directly from the unlabeled dataset so that we run the experiment on the test set without using the training dataset. The third method is Siamese models where we learn the model on the training set and use it to extract the features on the test set then clustering them by k-means++. We use the backbone of CNNs for both DAC and Siamese models. DAC and Siamese consistently outperform the baseline of k-means+PCA due to the learned deep features. HSC features yield the best purity for all the datasets. The result confirms the effectiveness of spatial classification features over the deep features of the whole image.

We also evaluated the CNN-based feature on Dset\_Mix and compared it with the Weighted\_Comb feature. CNN-based feature achieved purity of 0.72 while the Weighted\_Comb feature's one was 0.84. The performance of the CNN-based feature was lower than the Weighted\_Comb feature. The reason is Dset\_Mix is synthesized from the CROHME 2016 dataset. We applied some methods of symbol normalization, translation, root stretch and the Convolutional Neural Network (CNN) is sensitive to those transformations. Hence, the network cannot detect the information of symbols and positions in the HME images effectively.

#### 3.8.4.2 *Effect of multi-level spatial features*

We evaluate the clustering performance for all three datasets with different configurations of multi-level spatial pooling. The results are shown in Table 10.

Table 10. Clustering results of different configurations of multi-level spatial pooling (purity)

	Spatial Pooling sizes				Datasets			
	1x1	3x3	5x5	7x7	HME_Answer_1	HME_Answer_2	CROHME_2016	CROHME_2019
Classification features	✓				<b>0.88</b>	0.95	0.92	0.86
Single-level HSC features		✓			0.75	0.93	0.96	0.93
			✓		0.73	0.92	0.97	0.92
				✓	0.70	0.91	0.96	0.92
Multi-level HSC features	✓	✓			0.85	0.98	<b>0.99</b>	0.95
	✓		✓		0.83	<b>0.98</b>	0.98	<b>0.96</b>
	✓			✓	0.80	0.97	0.98	0.93
	✓	✓	✓		0.80	0.97	0.98	0.96
	✓		✓	✓	0.79	0.97	0.98	0.93
	✓	✓	✓	✓	0.77	0.97	0.98	0.95
	✓	✓	✓	✓				



First, we confirm the effectiveness of localization information in HSC features. The single-level HSC features with spatial information (spatial sizes different with (1x1)) yield the purity of 0.99 and 0.96 on CROHME 2016 and CROHME 2019, outperform the HSC feature without spatial information where the purity are 0.92 and 0.86. For Dset\_22Qs and Dset\_50, however, spatial information does not show its effectiveness. The reason is that in both Dset\_22Qs and Dset\_50 the answers are composed of different symbol sets. Thus, classification features are enough for clustering them. On the other hand, varying writing styles make purity by large-sized spatial pooling HSC features worse than the small-sized spatial pooling HSC features. For CROHME, there are formulas with a similar set of symbols but different on structures, thus, classification information is not enough for clustering them.

Second, we confirm the effectiveness of combining multi-level HSC features over single-level HSC features. For both CROHME and Dset\_50, multi-level HSC features yield higher purity than single-level features. The best combination in CROHME 2016 is {1x1, 3x5} with purity of 0.99, the best in Dset\_50 and CROHME 2019 is {1x1, 3x7} with purity of 0.98 and 0.96. These are the results in Table 10. Combining more than three levels of HSC features is not as effective as purity is reduced. This is due to there are a lot of redundant information in the combined HSC features.

#### 3.8.4.3 *Effect of attentive pooling*

We measure the clustering performance on CROHME 2019 of the network which is trained with GAtP, GMP, and GAP, respectively. The clustering performance in purity is shown in Table 11. The network with GAtP yields slightly better purity than GMP and outperform GAP by a large margin when clustering with large-sized single-level HSC features. For clustering with multi-level HSC features, the network with GAtP also yields better results than GMP and GAP. The results suggest that GAtP produce better classification and localization than GMP and GAP. The network with GAP yields low purity of 0.28 when only classification features are used. Although the poor performance of classification features, HSC features with localization information still produce high purity of over 0.87 for clustering.

We analyze the classification prediction of the networks on CROHME 2019 by mean average precision (mAP). First, the per-class performance is measured using average precision by calculating the area under the precision-recall curve. The precision and recall of classification performance are for evaluating if a class exists in the HMEs or not. The average precision across all classes is summarized to calculate mAP. The mAP by the networks with GAtP, GMP, and

GAP is 0.93, 0.92, and 0.17, respectively. The results confirm that the network with GAtP yields better classification performance than GMP and GAP.

Table 11. Effect of global pooling methods for clustering (purity) on CROHME 2019

	Spatial Pooling sizes				Global Pooling methods		
	1x1	3x3	5x5	7x7	GAtP	GMP	GAP
<i>Classification features</i>	✓				0.86	0.82	0.28
<i>Single-level HSC features</i>		✓			0.93	0.93	0.74
			✓		0.92	0.92	0.77
				✓	0.92	0.91	<b>0.81</b>
<i>Multi-level HSC features</i>	✓	✓			0.95	0.94	0.76
	✓		✓		<b>0.96</b>	<b>0.95</b>	0.80
	✓			✓	0.93	0.93	0.80

#### 3.8.4.4 Effect of multi-scale feature extraction

We make the experiments to confirm the effectiveness of multi-scale feature extraction. We train the proposed networks to obtain the HSC feature by 1-scale (original input), 2-scale (original input, 1/2 scaled input), and 3-scale (original input, 1/2 scaled input, 1/4 scaled input). The results of clustering on CROHME 2016 and CROHME 2019 are presented in Table 6. The result confirms the effectiveness of multi-scale HSC features. HSC feature with 3-scale achieves the highest purity as compare with 1-scale or 2-scale HSC feature. The 2-scale HSC features also yield high purity of 0.95 on both the datasets. This suggests that 2-scale features could cover almost the scale of symbols in HMEs.

Table 12. Effect of multi-scale HSC feature (purity)

	CROHME 2016	CROHME 2019
HSC feature (1-scale)	0.77	0.75
HSC feature (2-scale)	0.95	0.95
HSC feature (3-scale)	<b>0.99</b>	<b>0.96</b>

#### 3.8.4.5 Visualizing class activation maps

The class activation map for an HME is shown in Figure 23. We extract the output of CNN for multiple scales of input images as presented in Figure 13. We resize all the outputs to the original image size and obtain the combined multi-scale output by taking maximum values across the output images. The activation map for all the classes is also presented by taking the maximum output across the classes.

From the visualization, the symbols in different sizes are handled correctly by the three scales of input images. Small size symbols such as ‘*n*’, ‘*j*’, ‘*x*’, ‘*a*’ in the formula are detected by the first scale, medium size symbols such as ‘(, ’)’, ‘=’, ‘*P*’, ‘sum’ are detected by the second scale, only

symbol ‘sum’ is detected by the third scale. The locations of symbols detected by the networks are also correctly predicted. There are cases in which both ‘(’ and ‘)’ are located in the same position. Since ‘(’ and ‘)’ are always presented in pairs in ME, there is no information to discriminate them by using only weakly supervised label.

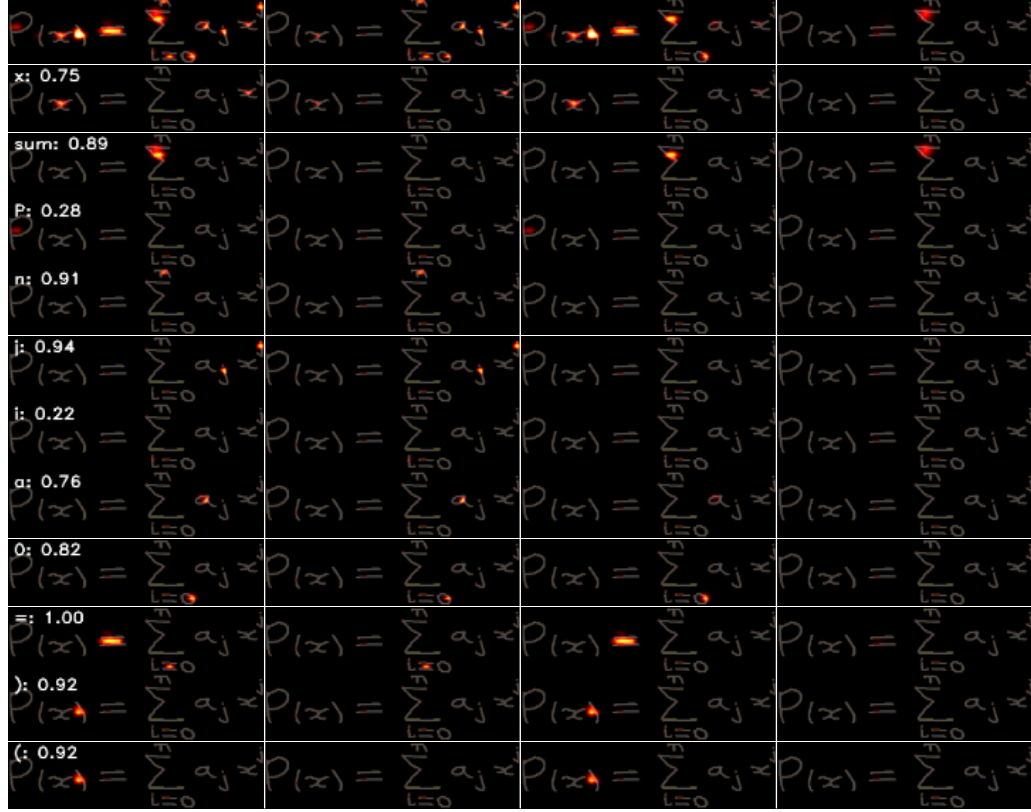


Figure 23. CAMs by multi-scale CNN with attentive pooling. The columns from left to right are the aggregated of multi-scale CAMs with class label and predictive score, the first, the second and the third level CAMs, respectively. The first row shows the combination CAMs results for all the classes.

Generally, model to learn both class and location representations of symbols for clustering HME. The networks could detect symbols in various scales by a combination of features from a multi-scale CNN. Training the networks with a combination of global attentive pooling and global max pooling improves classification and location prediction. Clustering by multi-level spatial representations extracted from CNN prediction outperforms the online and offline Bag of Features by a large margin. The method also achieves high performance (purity of 0.99) for the CROHME dataset with 36 clusters.

A drawback of the method is that it is not feasible to group the answers with multiple mathematical representations. For example, the formula “1/2” could be represented as “0.5”, “1/2”, “1÷2”, etc. This may need the recognition of the whole formula as a post-processing step.

## 4 Synthetic Data Generation

### 4.1 Introduction

Although offline recognition of printed MEs has been improved and there are commercial and open-source programs that can achieve very good results, recognition of HMEs recognition is still an open problem, because the variability of the structures, styles, and the deformations of the symbols present some challenging problems.

HME recognition has been an intensive research field for many decades. Besides the recognition algorithms and the types of features used, the amount and quality of training data also have a great impact on the recognition performance of handwriting recognition methods. As a rule of thumb says, the classifier trained on the most data wins. This has been experimentally justified in many works before. In recent years, deep learning-based HME recognition methods have managed to achieve very good results, but they need an extensive dataset to be trained with.

The most straightforward way to expand the training set would be to collect additional natural, i.e. human written, samples. However, the production of these datasets is costly, because it is a rather expensive and time-consuming process. Some methods have been developed to mitigate this cost by distributing the work amongst a large number of people, such as crowdsourcing [74], but it is still not an ideal solution.

The most cost-effective method would be to artificially generate labeled samples. This would allow us to have an effectively infinite dataset, without any human labor required. Thus, developing a synthetic handwritten text generator is very attractive. Ideally, a good data generation strategy can enlarge a dataset even with limited amounts of synthetic samples.

A lot of works have been done in the field of synthetic text generation that can classify into three main approaches: data augmentation and texture synthesis approach [75] [76], font-based approach [77], and generational approach [78] [79].

While the synthetic text only is generated based on one-dimension structure language (left to right, top to bot, or right to left), synthetic HMEs are more complex since mathematics is a two-dimension structure language. Just a few approaches have been presented for generating the synthetic HMEs. For the symbol level, Davila et al. [80] generate online synthetic mathematical symbols using an elastic distortion model. For the expression level, Awal et al. [81] generate online pseudo-synthetic HMEs using a stochastic layout guided by the LaTeX string defining the expression, but they only use a single handwritten pattern for each symbol. Le et al. [82] apply

local and global distortions models and generate new patterns based on the real existing online HME patterns, but this method cannot generate a new pattern from a mathematical description language such as LaTeX and Mathematical Markup Language (MathML).

Since the mathematical structure is very complex, we introduce a data augmentation and texture synthesizer which can generate new, automatically labeled HME patterns, combining the layouts and isolated symbols of the supplied dataset and applying some random methods to increase the realism and variability. To evaluate the synthesizer, we conduct a questionnaire-based experiment for the human subjects (volunteers) to rate the clarity and the naturalness of generated synthetic HME patterns.

To summarize, we make two novel contributions. Firstly, we can generate a completed synthetic HMEs from any given LaTeX or MathML script. Second, our synthesizer can produce realistic HMEs in a wide variety of structures and styles.

## 4.2 Synthesizer

The main objective of this project is to create a synthesizer capable of taking a plain LaTeX or MathML script as the input and producing human-like HME patterns as the output, together with its corresponding ground-truth at any desired level. To accomplish this task, we need to consider several separated sub-tasks. A high-level scheme of the generation process is shown in Figure 24.

We focus on generating online patterns since the future ultimate objective would be to improve the recognition rate of online HME recognition. The offline patterns are also generated by rendering the online patterns into bitmap images. The details of our synthesizer are described in the subsequent sections.

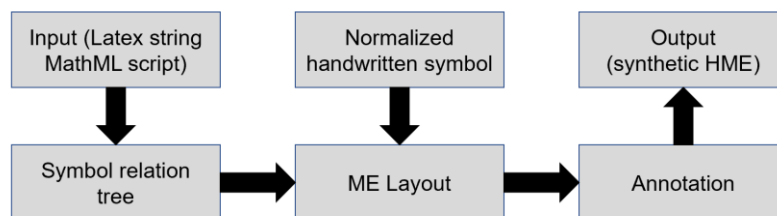


Figure 24. The pipeline of the synthesizer.

### 4.2.1 Input

Our synthesizer accepts two kinds of input: LaTeX and MathML script. These are two of the most popular formats which can represent MEs in digital form. Whereas LaTeX is a digital

description language for storing and presenting MEs, MathML is a mathematical markup language based on XML for describing mathematical notations and capturing both its structure and content.

The synthesizer also requires a sample dataset of handwritten mathematical symbols. If we randomly pick up several patterns in a single symbol category, very often we have more than one writing style of that symbol. Since each person has a different writing style and tends to use a unique symbol style while writing, a preprocessing step needs to be applied to the input dataset that clusters the writing styles of each symbol into several style groups. Although the clustering task for each category costs effort and time, it is necessary to make the synthetic patterns more natural.

#### **4.2.2 Symbol Relation Tree**

The synthesizer converts the LaTeX and MathML script into a symbol relation tree (SRT) which represents spatial relationships between symbols more obviously. In an SRT, nodes represent symbols, while labels on the edges indicate the relationships between symbols. We consider the first appeared symbol in the LaTeX or MathML script as the root node of the SRT. As the relations between the symbols are inherited from the math semantics, the graph built with the symbols and their relations is a tree. In an SRT, one node (except the root node) connects directly with only one parent node.

In our system, we classify the relation into six groups: horizontal adjacent, subscript, superscript, above, below, and inside.

#### **4.2.3 The layout of the Synthetic HME**

The layout of the synthetic HME is defined as a set of labeled points on the Cartesian coordinates' system. To determine the layout of the synthetic HME, the two most important elements are the position and the size of each symbol in the SRT. These two elements of a symbol can be defined by the coordinators of four points of its bounding box which are top, bottom, left, and right.

Since the sizes of mathematical symbols are different even in printed symbols, we divide the symbols into several groups based on two criteria. One is the relative position of the printed style of symbols with the baseline and the mean line. There are four kinds of relative positions: normal (NOR), ascendant (ASC), descendant (DES), and small (SMA). NOR symbols fall between two

lines; SMA symbols are similar to the NOR symbols but have a very small size; ASC and DES symbols extend above the mean line and below the baseline, respectively. We also extract the symbol's body box that only covers the main body of the symbol like the work of Le et al. [83]. Another criterion is the ratio between the width and height of the bounding box of each symbol in the printed style. Table 13 shows an example of grouping 101 symbols in the CROHME 2016 dataset.

Table 13. 12 Groups of symbols in CROHME 2016 dataset

#	Criteria		Symbols
	Position	Size	
1	NOR	Height $\approx$ Width	a, c, e, i, n, o, r, s, u, v, x, z, $\backslash$ alpha, $\backslash$ phi, $\backslash$ pi, $\backslash$ sigma, $\backslash$ times, +, =, $\backslash$ div, >, <, geq, leq, $\backslash$ in, $\backslash$ neq, $\backslash$ pm
2	NOR	Height < Width	m, w, $\backslash$ infty, $\backslash$ rightarrow
3	NOR	Height $\ll$ Width	$\backslash$ sin, $\backslash$ cos, $\backslash$ tan
4	NOR	Width is varied	- (or fraction bar)
5	ASC	Height > Width	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, b, d, f, h, k, l, t, A, B, C, D, E, F, G, H, I, L, M, N, P, R, S, T, V, X, Y, $\backslash$ beta, $\backslash$ Delta, $\backslash$ exist, $\backslash$ forall, $\backslash$ lamda, $\backslash$ theta, /
6	ASC	Height $\ll$ Width	$\backslash$ lim
7	ASC	Height $\gg$ Width	!,  , (, ), [, ], {, }
8	DES	Height > Width	g, p, q, y, $\backslash$ gamma, $\backslash$ mu
9	SMA	Height $\approx$ Width	$\backslash$ comma, $\backslash$ dot, $\backslash$ prime, $\backslash$ ldots (split into 3 $\backslash$ dot)
10	ASC & DES	Height > Width	$\backslash$ int, $\backslash$ sum
11	ASC & DES	Height $\ll$ Width	$\backslash$ log
12	ASC & DES	Height and width are varied	$\backslash$ sqrt

The layout is initialized by specifying the size and position of the symbol in the root node of the SRT. The default initialized height and bottom-left position of the body box are 100 pixels and (0, 0), respectively, which are modified based on the generating purpose. The baseline and the mean line are defined as the lines paralleled to the x-axis that pass the bottom and the top points of the body box, respectively. The sizes and positions of the remaining symbols are determined based on that of the root symbol and the relations.

The order of considered symbols is determined according to the Depth First Search (DFS) algorithm. The algorithm starts at the root node and explores as far as possible along each branch before backtracking. The size and position of the symbol in the child node (child symbol) are based on its relation with the symbol in its parent node (parent symbol), as shown in Table 14. Where “s\_ratio” is the percentage between the size of the child symbol (“sc”) and that of the

parent symbol; “s\_point” (setpoint) is the initialized point of the bounding box of the child symbol; “r\_point” (reference point) is the point of the bounding box of the parent symbol used to determine the “s\_point”; “dx” and “dy” is the absolute distance between “r\_point” and “s\_point” on x-axis and y-axis, respectively.

Table 14. Size and position of the child symbol based on the parent symbol

Relation	s_ratio	r_point	s_point	dx	dy
Horizontal	100%	Bot-right	Bot-left	sc/5	0
Subscript	40%	Bot-right	Top-left	sc/10	cs/3
Superscript	40%	Top-right	Bot-left	sc/10	cs/3
Above	30~70%	Top-left	Bot-left	0	cs/5
Below	30~70%	Bot-Left	Top-left	0	cs/5

In an expression contained above and/or below relations such as  $\lim_{x \rightarrow \infty} \frac{1}{1-x}$ , the synthetic HME pattern will be more natural if we align the above or below sub-expression based on its parent symbol (e.g. \lim, fraction bar). The alignment should be done after finishing positioning all symbols in the sub-expression because it depends on the total width of the sub-expression. In the fraction case, the width of the fraction bar will be determined after the sizes of both the numerator and denominator are available.

The creating layout process is repeated until the DFS algorithm reaches the last node in the SRT. The layout, however, looks like printed patterns due to the fixed parameters. To widen the variety of the layout, the value of every parameter is perturbed from -15% to 15% by multiplying a random value.

#### 4.2.4 Selecting and Positioning Handwritten Symbols

In practice, people often tend to use the same handwriting style for each kind of symbol. We suppose that each symbol class is already clustered in the input dataset. If any symbol appears more than one time in the input, our synthesizer will select different handwritten symbol patterns but in the same style. To expand the variety of the writing style for the synthetic HME patterns, the synthesizer randomly selects the handwriting style for symbols appearing in the input for each time of HME generation.

The synthesizer then assigns each selected handwritten symbol pattern to a corresponding bounding box in the generated layout. The size of the corresponding bounding box, however, is usually different from that of the bounding box of the selected pattern. Hence, the synthesizer normalizes all symbol patterns before positioning them into the corresponding places in the



layout. The normalization method only aims to resize the handwritten symbol patterns without changing their original ratio between height and width.

#### 4.2.5 Root Symbol with inside relation

The root symbol is a special case since its height and width depend on the sub-expression inside it. When its width increases or decreases, only the size of the overline part is extended or shortened. On the other hand, its radical part should be adjusted to change when the height of the root is altered. Therefore, the handwritten root symbol must have a different normalization method from the other symbols.

Based on this analysis, we must divide the handwritten root symbol into the overline and the radical part before normalizing it. Figure 25 shows an example of processing a handwritten root pattern where points A and D are the beginning and ending points of the patterns, respectively. Firstly, we reorder and connect all the strokes into a single stroke if the original pattern has more than one stroke. Second, we aim to detect the point B corresponding to the global valley point in the printed root symbol. Figure 25 (a) shows an example of point B in a handwritten root pattern. In the handwritten root patterns, however, the global valley point sometimes locates the beginning or ending part. Hence, we apply the local valley detection algorithm to locate all the local valley points. Then, we select the point B as the point in the middle part of the pattern. Third, we locate point C that divides the pattern into two parts by applying the following simple algorithm. The algorithm considers each segmented line of a polyline formed from point B to point D of the pattern. The first segmented line whose angle formed with the x-axis is less than 20 degrees, the beginning point of that line is detected as point C. Figure 25 (b), point  $B_1$  is considered as point C. Finally, we divide the original pattern into two parts at point C and normalize each part according to the inside sub-expression. Figure 25 (c) shows the normalized radical part and overline part for the sub-expression  $1 + z^2$ .

After the normalized root pattern is placed into the layout, its corresponding sub-expression will be positioned below its overline part and on the right side of its radical part, as shown in Figure 25 (d).

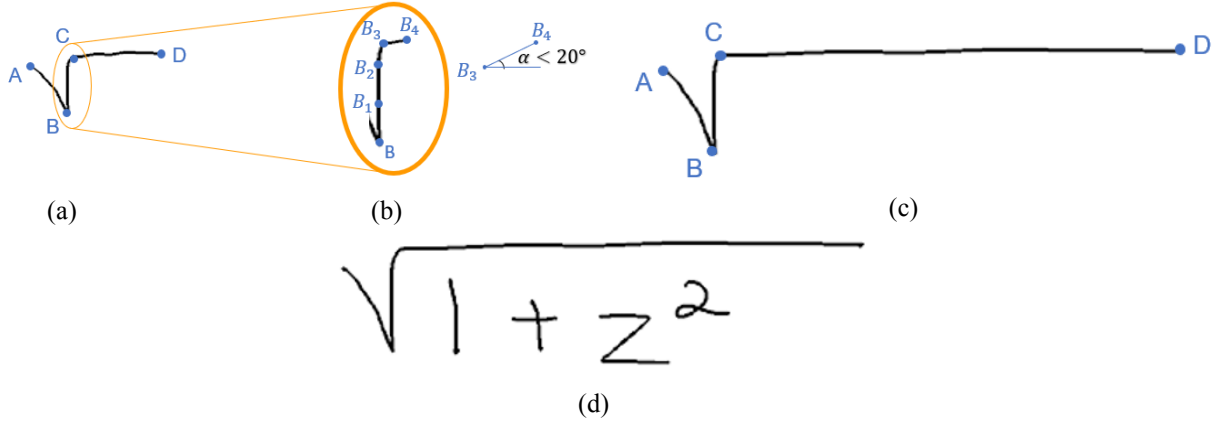


Figure 25. An example of generating the root symbol with the inside sub-expression.

#### 4.2.6 Output

Throughout the synthetic process, the synthesizer keeps and links all the information of input symbols and strokes. Therefore, it can automatically make annotations and create online synthetic HME patterns in InkML format. The offline patterns are also created by rendering the online patterns to the bitmap image format. The ground-truth of each synthetic HME pattern is also attached in both LaTeX and MathML format.

### 4.3 Experiments

A group of 30 human subjects (volunteers), consisting of people of both sexes and different age groups with varied educational backgrounds, were invited to answer our questionnaire-based experiments. All the volunteers regularly use HMEs for either studying or working purposes.

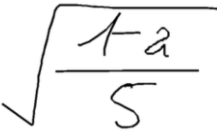
Totally 500 synthetic HME patterns were generated by the proposed method. To make the experiments that can cover a wide range of contexts of mathematics, we classify these synthetic patterns and the real patterns in the CROHME 2016 testing dataset into eleven categories. Then, at least one synthesis pattern and one real pattern are randomly picked up from each category. Table 15 shows the number of picked-up patterns in each category and Figure 26 shows four synthetic patterns that are picked up for being evaluated in the experiments. The first seven types are considered as simple patterns since there are only a few symbols and relations in each pattern. Patterns in the remaining types are complex patterns that contain a lot of symbols and relations.

For volunteers to do the experiments easier, all the patterns were rendered into the bitmap images since ordinary people usually read HMEs in the offline format. The order of 50 patterns was also shuffled before showing it to the volunteers. The following three questions were asked for every pattern:

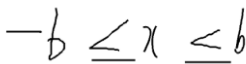
- Give the correct ground-truth of the pattern.
- Evaluate the naturalness of the pattern.
- Point out the unnatural points (if existing) in the pattern.

Table 15. Number of each kind of patterns in the experiments

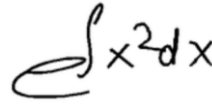
#	Expression types	# synthetic patterns	# real patterns	Total
1	Root	4	2	6
2	Letters	5	1	6
3	Numbers	5	1	6
4	Subscript	3	1	4
5	Superscript	5	1	6
6	Fraction	4	1	5
7	Big symbols	2	2	4
8	Complex fraction	2	2	4
9	Complex relation	2	1	3
10	Complex root	2	2	4
11	Long expression	1	1	2
<b>Total</b>		<b>35</b>	<b>15</b>	<b>50</b>



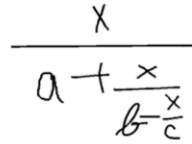
(a)



(b)



(c)



(d)

Figure 26. Four examples of synthetic patterns in the experiments.

#### 4.4 Study of the clarity of the synthetic samples

This experiment was designed to statistically explore the clarity of the synthetic samples compared to the real samples. A clarity sample is an HME whose symbols and relations are readable and recognizable for ordinary people. Datasheets with both synthetic and real samples were shown to the volunteers who were asked to write down the ground-truth of all samples in LaTeX format or math editors. Then we compared the volunteer ground-truth with the original one and calculate the recognition rates for synthetic patterns and real patterns.

The results show that 85.94% of synthetic samples are recognized correctly while the rate for the real samples is 88.33%. We tested the hypothesis that the clarity of the synthetic samples is similar to that of the real ones. A two-sample t-test was computed with the level of a significant 5%. Thus, we conclude that the synthetic samples are as clear as the real samples ( $t = -0.36$ ,  $p\text{-value} = 0.72$ ).

The misrecognition may be caused by many factors. Subjective factors come from the writing style or generation method whose symbol or relation hard to recognize. On the other hand, objective factors are usually caused by missing context information.

#### 4.5 Study of the naturalness of the synthetic samples

We designed the following experiments to evaluate the naturalness of synthetic samples to human subjective perception. The volunteers rated the naturalness in the 5-point scale, from 1: looking strongly like synthetic to 5: looking strongly like real.

Table 16 shows detailed results of mean scores and standard deviations of each kind of sample. The mean scores of the naturalness of simple patterns are lower than that of the complex patterns in all three columns. The reason is that even the small unnatural points are easy to be recognized by human subjects in simple patterns. On the contrary, the human subjects took an overview looking at the complex patterns and tend to ignore small unnatural points, only big ones could be detected.

Table 16. Mean scores and Standard deviation for each kind of patterns

	Synthetic patterns	Real Patterns	Synthetic & real patterns
<b>Simple patterns</b>	$3.46 \pm 0.60$	$3.85 \pm 0.62$	$3.55 \pm 0.62$
<b>Complex patterns</b>	$3.78 \pm 0.51$	$3.92 \pm 0.65$	$3.84 \pm 0.56$
<b>Simple &amp; complex patterns</b>	$3.53 \pm 0.59$	$3.88 \pm 0.61$	$3.63 \pm 0.61$

The results in Table 16 also show that the mean scores of the synthetic patterns are slightly less than that of the real pattern in all kinds of simple and complex samples. These small gaps empirically establish that the synthetic samples look as natural as the real ones. Indeed, when synthetic samples mixed with the real samples are presented to human subjects, the subjects are highly confused in identifying the synthetic ones due to their naturalness.

If a synthetic sample does not have a significant abnormal point, then a human subject should judge it as natural as the real one. Thus, the difference of naturalness between the synthetic patterns and the real ones was compared by using a two-sample t-test. The result shows that there is no significant difference of the naturalness between them ( $t = -1.87$ ,  $p\text{-value} = 0.07 > 0.05$ ).

#### 4.6 Study of the unnatural points in the synthetic samples

We have shown so far that the clarity and naturalness of the synthetic samples nearly reach that of the real samples, and thus, the proposed strategy is capable of increasing the number of HME samples for a training dataset. The experiment was designed to collect the comments of unnatural points for each pattern. Our present goal is to evaluate the proposed method concerning

its capacity to generate synthetic samples. Therefore, we only considered the evaluation of synthetic patterns and ignored the real ones.

There are 48% of synthetic patterns that got at least 1 comment. These comments can be classified into five general problems: (A) Inconsistent writing style of symbols, (B) Symbol size is too large or small, (C) Ambiguous relation between symbols, (D) Strange or weird writing style of symbols, and (E) Unnatural distance between symbols. Figure 27 shows the percentage of each problem in the commented patterns. These problems are mainly caused by the random parameters, the symbol normalization method, and the choice of the real patterns in the isolated symbol dataset.

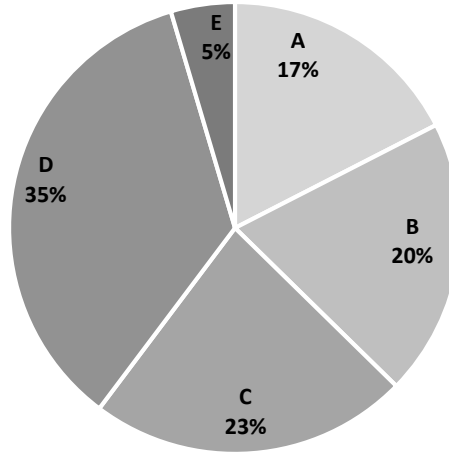


Figure 27. Five common unnatural reasons for synthetic patterns.

The experimental results prove that people cannot distinguish the real patterns and synthetic patterns. On the other hand, the addition of synthetic HMEs to the natural training set may improve the recognition rate, but the proportion of real and synthetic HMEs plays an important role.

These results suggest that additional work should be conducted in the future to make the synthetic HMEs represent more diverse writing styles. For example, increasing the quantity and quality of isolated symbols would be a straightforward step towards this direction. Besides, clustering symbols into specific styles and sizes would allow the synthetic HME patterns to be much more human-like. Furthermore, different writing styles could be emulated by tuning the generator's parameters.

## 5 Interactive User Interface for Inputting MEs

### 5.1 Introduction

Among dozens of handwriting recognition problems, HMER not only shares some common challenges such as ambiguous handwriting input, variant scales of symbols, etc. but also characterized by the complicated two-dimensional structures. For example, in the Handwritten English Recognition, the handwritten characters are arranged sequentially from left to right according to each line, while for HMER, the handwritten characters are arranged on a two-dimensional layout according to relations among them (e.g. above, below, horizontal, fraction, inside). The output of the HMER can be represented by a variety of formats such as LaTeX, MathML, etc.

Since mathematics is widely used in education, science, business, engineering, medicine, economics, etc., it will be very effective if the users can input mathematical expressions naturally using handwriting. Generally, there are two approaches to handwriting recognition. PDAs (Personal Digital Assistants), tablet PCs, and electronic whiteboards require online (real-time sequences of pen-tip/finger-top coordinates) recognition whereas scanned copies of handwritten documents require offline (images) recognition.

Currently, there are three primary methods for entering mathematical expressions (MEs) into computers, by mathematical script languages, by mathematical formula editors, and by computer recognition of handwritten mathematical expressions (HMEs). The mathematical script languages require the users to be familiar with a programming language (e.g., LaTeX) to input MEs. Meanwhile, the mathematical formula editor (e.g., MathType) often forces the users to select predefined mathematical structure templates (e.g., fractions, superscripts, subscripts) from a menu or toolbar and then fill in the templates with numbers, letters, and so forth by typing on the keyboard. These two methods, however, have difficulties for students to use in the learning process, especially in answering descriptive questions.

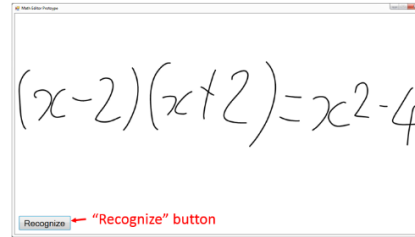
On the other hand, the HME recognition-based method is a natural choice for inputting MEs. However, this method requires a highly reliable HMEs recognizer. HMEs are more challenging to recognize than natural languages since the structural ambiguity is higher. In fact, the recognizer of HMEs usually has problems with recognition errors. The errors could come from various levels of recognition: symbol segmentation, symbol recognition, and structure analysis. By providing a

user interface with which the user can quickly correct recognition errors, the user experience could be significantly improved.

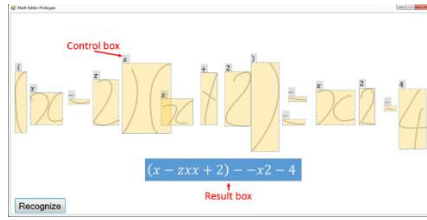
In this research, we prototype and evaluate two versions of the user interface for representing the HMEs recognition result. Moreover, we introduce various gestures for editing recognition errors. We also conduct a user study to evaluate and compare them to each other.

## 5.2 User interface design for displaying recognition result and misrecognition correction

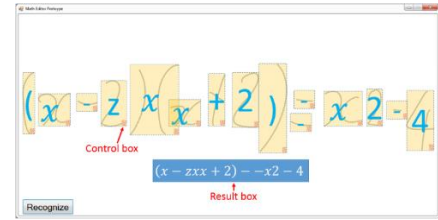
Usually, users do not know where misrecognitions come from. Hence, we aim to design a user interface that represents the HMEs recognition result such that the users perceive the recognition result and identify the recognition errors quickly. First, our user interface starts with an empty writing panel and a “Recognize” button. When the users complete their writing MEs, they can press the “Recognize” button to “tell” the system to recognize the entire HME (Figure 28 (a)).



(a) Initial user interface.



(b) UI\_on shows symbol recognition results on its bounding box.



(c) UI\_inside shows symbol recognition result within its bounding box.

Figure 28. Two UIs for showing and editing HMEs recognition result.

After pressing the “Recognize” button, our user interface shows the recognition result in the result box below the HME and enables the users to edit recognition errors. Moreover, it shows the symbol segmentation using bounding boxes. To show the recognition result of each symbol, we consider two versions. The first version shows the symbol recognition result at the left position on its bounding box, as shown in Figure 28 (b) (called UI\_on). The second version embeds the symbol recognition result inside the bounding box, as shown in Figure 28 (c) (UI\_inside). The control box is associated with the bounding box, which captures some editing gestures.

### 5.3 Editing Gestures

There are recognition errors in symbol segmentation, symbol recognition, and structure analysis. For each type of error, we propose a couple of gestures for correcting recognition errors.

#### 5.3.1 Gestures for Editing Symbol Segmentation Errors

The initial step in HME recognition is the symbol segmentation process. The wrongly segmented components are likely misrecognized in the following steps of HME recognition. Hence, we propose two types of gestures to correct segmentation errors when the user identifies missegmented components. In our user interface, the user can recognize segmented symbols using their bounding boxes.

The segmentation errors can be divided into two major types (the others can be a mixture of them):

- Under-segmentation: A segmented component includes strokes belonging to multiple symbols.
- Over-segmentation: A segmented component includes a subset (but not the whole) of strokes of a single symbol.

The correction process of a segmentation error is triggered by the user to draw a gesture stroke on the missegmented segmentation components. To correct an under-segmentation error, the user draws a stroke starting from outside the bounding box of the under-segmented component, penetrating it by separating the component and ending outside as shown in Figure 29 (b).

On the other hand, for correcting an over-segmentation error, we propose two gestures: (i) a stroke starting from inside of the bounding box of an over-segmented component to the inside of that of another as shown in Figure 29 (c); and (ii) a stroke encircling the bounding boxes of over-segmented components as shown in Figure 29 (d). For a symbol over-segmented into 3 or more than 3, the user may apply the first gesture successively or apply the second gesture. The first is suitable when strokes are crowded while the second allows the user to combine all the over-segmented components by only a single action.

Gesture (b) and gesture (c) are effective for highly packed groups of symbols, while gesture (d) is naïve and efficient for sparse symbols.

To give feedback to the user's gesture, the segmentation and symbol recognition result is updated immediately after the correction gesture is finished.



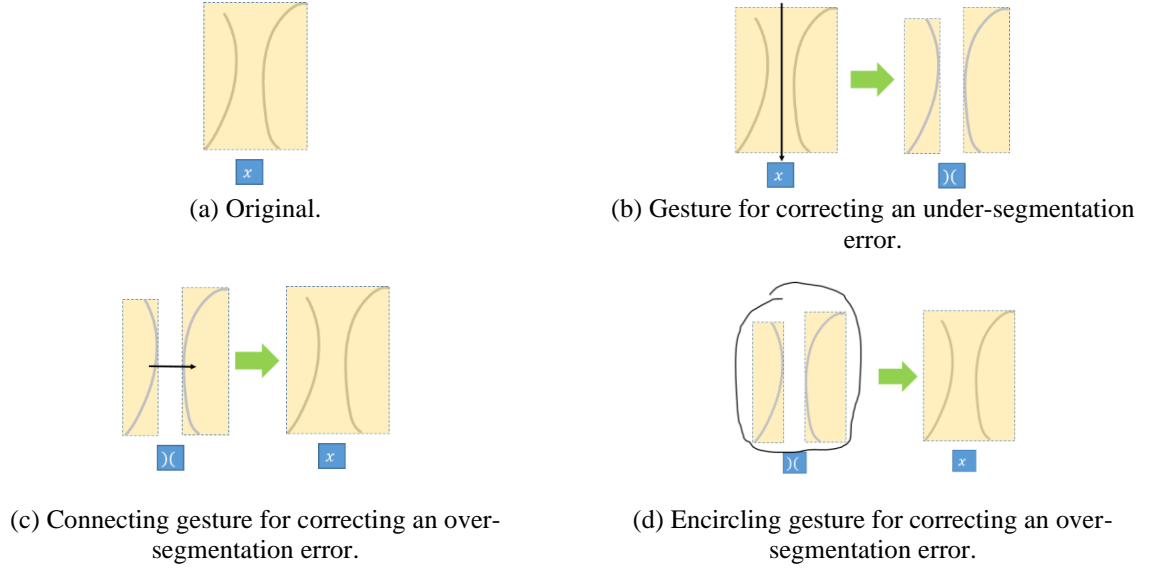


Figure 29. Gestures for correcting symbol segmentation errors.

### 5.3.2 Gestures for Editing Symbol Recognition Errors

After segmenting symbols correctly, the symbol recognition process still may misrecognize the segmented components. Hence, we allow the user to correct the errors by interacting with the control boxes. We propose three approaches to handle this problem:

- Single-tap: In the first approach, a symbol list menu is shown by tapping the control box. The system displays a list of top N-candidates based on the recognition scores and the “Other” option. If the expected symbol is not in the candidate list, the user could access the full list of symbols by tapping the “Other” option followed by several groups and finally the expected symbol as shown in Figure 30 (b).
- Multiple-tap: In the second approach, the users could switch among the recognition candidates by tapping the control box a few times. Then, tapping one more time shows the full list of symbols to allow the user to select the expected result, as shown in Figure 30 (c).
- Multiple-and-long-tap: In the third approach, similar to the second approach, the recognition candidates are also switched by tapping the control box. However, the full list of symbols is accessed by a long tap gesture.

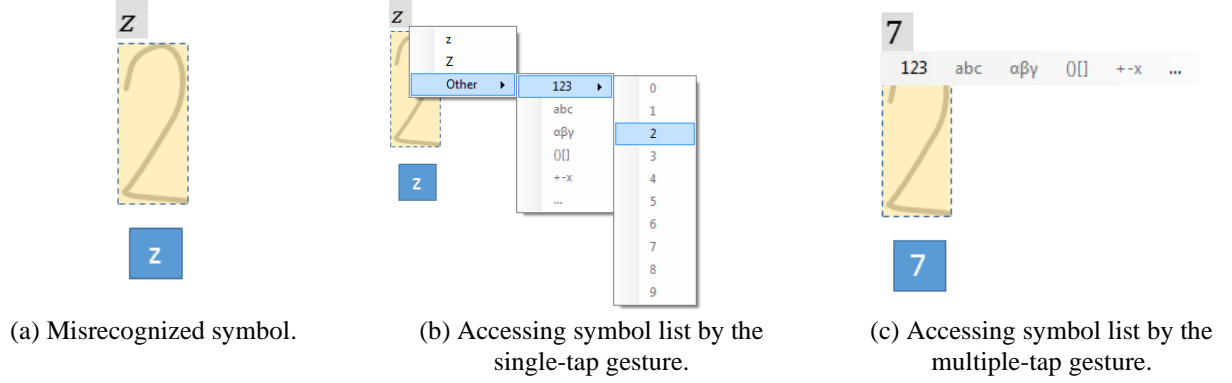


Figure 30. Symbol list for editing symbol misrecognition by gestures.

### 5.3.3 Gestures for Editing Structure Analysis Errors

This gesture is to correct misrecognitions in the structure analysis process. They occur at relations between two symbols. One symbol is the base of the relation, and the other is a leaf. For example, in the superscript relation of “ $x^2$ ”, “ $x$ ” is the base symbol while “2” is the leaf symbol. The structure analysis is reflected in positions and sizes of symbols in the recognition result. To trigger the process of correcting a structural relation error, the user taps on the control box of the leaf symbol and drags it into the effective region of the base symbol as shown in Figure 31.

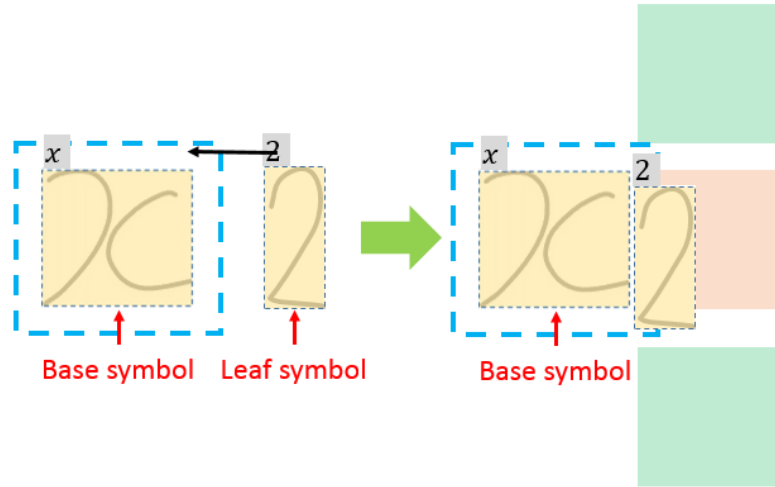


Figure 31. Effective region of base symbol “ $x$ ” enclosed by the dashed blue rectangle

We propose two approaches: recognition-based gesture and non-recognition-based gesture. In the recognition-based gesture, after dragging the leaf symbol into the effective region of the base symbol, relational boxes are displayed such that each relational box corresponding to one of the

following structural relations: horizontal, superscript, subscript, over, under, inside. The box corresponding to the current recognition result is highlighted in red, while the others are in green. The user drags the misrecognized symbol and drops it to the relational box corresponding to his/her expected structural relation. Based on the recognition result, the relational boxes are displayed according to the base symbol. If the base symbol is a fraction bar, the boxes of horizontal, over, under are displayed. If the base symbol is a root sign, the boxes of horizontal and inside are displayed. If the base symbol is a number or letter, the boxes of horizontal, superscript, and subscript are displayed as in Figure 32. Hence, unexpected structural relations are excluded.

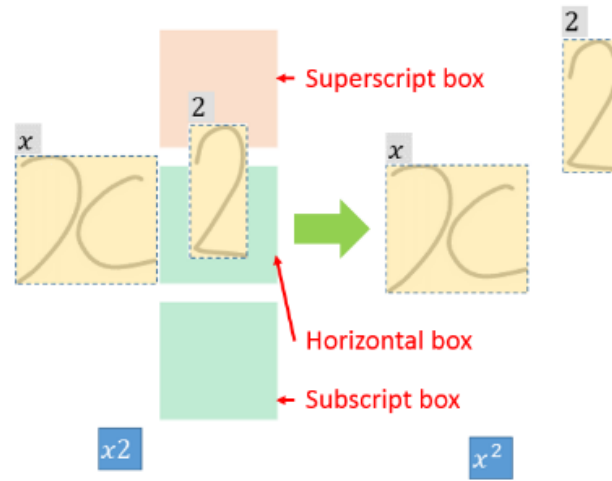


Figure 32. Recognition-based gesture for editing structure analysis

For the non-recognition-based gesture, the editing process is also triggered by dragging the leaf symbol to the base symbol's effective region. However, all the possible relational boxes are shown so that the user could arbitrarily modify the expressions as shown in Figure 33. Particularly, the user could drag the symbol he/she wants to modify and drop it to the desired box. Again, the system gives feedback by highlighting the selected box in red. After recognizing the gesture, the system will translate the leaf symbol and update the recognition result.

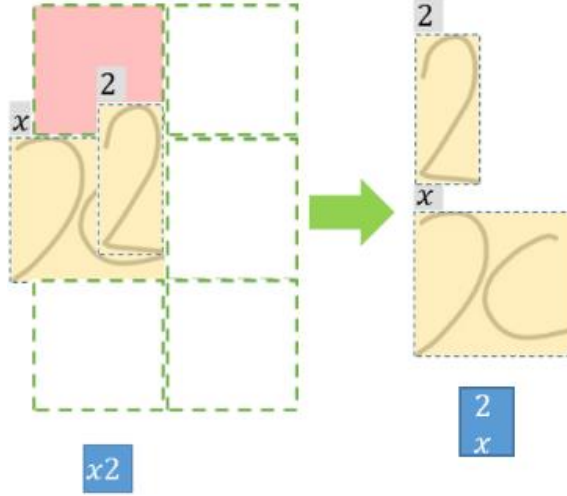


Figure 33. Non-recognition-based gesture for editing structure analysis

## 5.4 Experiments

Our experiment is divided into two steps: the first step is focusing on evaluating and comparing the effectiveness of the two user interface prototypes and the editing gestures while the second step is evaluating the editing experience when using the user interface prototype selected.

In the first step, we let the users try the two versions of the user interface and the editing gestures at the same time without any instruction or training. For the two versions of the user interface, we presented them at the same time and let the participants evaluate. For evaluating the editing gestures, we gave sample expressions that contained recognition errors and asked participants to correct them. After the users' try, we asked them to give their evaluation on the gestures even if they succeed or not. Most of the participants could imagine the correct gestures without explanation. Then, we explained the correct gestures when the user could not figure out how to perform the gestures. In the second step, we let the users try correcting the misrecognition of the HME in Figure 28 (a) on the user interface prototype selected based on their evaluation in the first step. The editing gestures are also selected according to their evaluation.



Figure 34. Sample pattern for evaluating a gesture to edit structural relation.

We recruited 20 participants. Their age ranged from 21 to 30 ( $25.7 \pm 2.66$ ). Fifteen participants were male and five participants were female. Eighteen participants were right-handed and two are left-handed. Five participants had never use pen-based or touch-based devices for writing, note-taking, and so on, whereas the others had some prior experience with the devices. The participants' backgrounds included information technology (16), construction (1), chemistry (1), biology (1), and mechanical engineering (1). This experiment was conducted on Surface pro3: Intel Core i5-4300U CPU (2.5 GHz) RAM 4Gb, Microsoft Windows 10 Professional.

#### **5.4.1 Comparing two interface versions and editing gestures**

To compare the two versions as well as compare the editing gestures, we employed the 5-point Likert Scale from 1 (strongly dissatisfied with the user interface or editing gesture) to 5 (strongly satisfied with the user interface or editing gesture). For each case, we collected the users' evaluation and obtained the average Likert points. First, we presented the two versions of the user interface, as shown in Figure 28 (b) and Figure 28 (c) on the same screen, and asked the participants to identify recognition errors. We asked whether the proposed user interface prototypes enabled them to perceive the recognition result and identify recognition errors easily. UI\_on achieved 3.85 points, whereas UI\_inside achieved 3.65 points. One of the main reasons UI\_inside was evaluated lower is due to the representation of the symbol recognition results. When the patterns are written densely, their bounding boxes tend to overlap with other ones, and so does the symbol recognition results. Moreover, the representation visually confuses the baseline of the expression because of the large character size. On the other hand, UI\_on shows the symbol recognition results apart from the handwritten patterns with small character sizes. Hence, it does not affect the handwritten pattern and its baseline as much as UI\_inside does. However, we still need to consider the size of recognized symbols for both versions.

For evaluating the gestures to correct symbol segmentation errors, we showed two handwritten samples as shown in Figure 29 (a), and asked the participants to use the gesture to separate the strokes to form “)(” pattern or to merge them to form the “x” pattern and evaluate each gesture. The users were supposed to perform the gesture in Figure 29 (b) for separating the strokes on both samples and perform the gestures described in Figure 29 (c) for merging one sample and the gestures described in Figure 29 (d) for merging the other. According to the participants' evaluation, the connecting gesture is significantly preferred with 3.95 points

(encircling gesture: 2.30). By applying paired samples t-test, we found that the connecting gesture was significantly evaluated higher than the encircling gesture ( $p\text{-value} < 3.1 \times 10^{-5} < 0.05$ ). While the connecting gesture only requires the gesture stroke to touch the over-segmented components, the encircling gesture requires the gesture stroke to encircle both the bounding boxes of over-segmented components.

For evaluating the gestures to correct symbol recognition errors, we showed three misrecognized handwritten samples, as shown in Figure 30 (a). Then, we asked the users to change all the symbol recognition to “2”. The users were supposed to figure out three different gestures described in Section 5.3.2 to correct the misrecognition and evaluate them. According to the participants’ evaluation, the single-tap gesture was dominant to the others with 3.70 points (multiple-tap gesture: 2.60, multiple-and-long-tap gesture: 2.70). We conducted two paired samples t-test: one with the hypothesis that the single-tap gesture was evaluated higher than the multiple-tap gesture (the p-values was less than  $5.2 \times 10^{-3} < 0.05$ ), one with the hypothesis that the single-tap gesture was evaluated higher than the multiple-and-long-tap gesture (the p-values was less than  $0.02 < 0.05$ ). Therefore, we concluded that the single-tap gesture was significantly evaluated highest. The reason is that the single-tap gesture could show multiple recognition candidates quickly by just one tap (and shows the symbol list as a context menu at the same time) while the others require multiple taps to switch to the next candidate and to access the symbol list. However, some participants could not find a way to correct the misrecognition in this task. The reason is that they tried to rewrite the misrecognized patterns again instead of interacting with the control boxes, which is not allowed in our prototypes. Hence, adding the rewriting function inside the bounding box would be necessary.

For evaluating the gestures to edit the structural relation, we showed two handwritten samples, as shown in Figure 34. Then we asked them to edit the relation between symbols to “ $x^2$ ” or “ $x_2$ ”. The user was supposed to figure out the gestures described in Section 5.3.3. According to the participants’ evaluation, the recognition-based gesture achieved an average of 3.60 points while the non-recognition-based gesture achieved that of 3.20 points. The recognition-based gesture is probably preferred since it could eliminate the inappropriate relational boxes and let the participants concentrate on the correcting process. However, there are some comments about the difficulty of interacting with the small control boxes. Enlarging the interacting area is an idea for improvement.

### 5.4.2 Evaluating user experience

After letting the participants experience the gestures, we let the user try correcting the expression in Figure 28 (a) on the user interface Figure 28 (b) or Figure 28 (c) based on their evaluation in the first experiment. We asked the users to edit the recognition result to “ $(x - 2)(x + 2) = x^2 - 4$ ”. Hence, the expression contains symbol recognition error (the first “2”), under-segmentation error (the “()”), over-segmentation error (two “-” of the “=”) and structural error (“x” and “2” on the right-hand side of “=”). Then, we asked a series of questions to obtain the participants’ evaluation of the prototype user interface. As shown in Table 17, the majority of users agree or strongly agree that the user interface enables them to correct the HMEs misrecognition as they expected.

Table 17. How do you rank the effectiveness (Can you edit recognition errors as you like)?

Extremely likely	5 (25%)
Likely	10 (50%)
Neither	4 (20%)
Unlikely	0 (00%)
Extremely unlikely	1 (05%)

Additionally, the majority of users agree or strongly agree that the user interface and the gestures are efficient, as shown in Table 18.

Table 18. How efficiently can you edit math expressions (How quickly, easily, and so on)?

Extremely efficient	4 (20%)
Efficient	10 (50%)
Neither	0 (00%)
Inefficient	5 (25%)
Extremely inefficient	1 (05%)

They also agree or strongly agree that the interactions with the user interface are clear and understandable, as shown in Table 19. Besides, the majority of participants say that it is easy to learn, as shown in Table 20. However, they thought it would be easier if there are instructions beforehand in general.

Table 19. Do you feel that your interaction with the user interface is clear and understandable?

Extremely likely	3 (15%)
Likely	10 (50%)
Neither	4 (20%)
Unlikely	3 (15%)
Extremely unlikely	0 (00%)

Overall, the majority of participants are satisfied with the user interface as shown in Table 21, and they preferred the designed user interface to the traditional math input user interface such as math editor and math language as shown in Table 22.

Table 20. Do you feel it is easy to learn to use the user interface?

Extremely likely	4 (20%)
Likely	10 (50%)
Neither	4 (20%)
Unlikely	2 (10%)
Extremely unlikely	0 (00%)

Table 21. How are you satisfied with the user interface?

Extremely satisfied	2 (10%)
Satisfied	11 (55%)
Neutral	6 (30%)
Dissatisfied	1 (05%)
Extremely dissatisfied	0 (00%)

Table 22. Which do you prefer among the recognize/edit math input, Math editor, and Math language for inputting math formula on the computer?

The recognize/edit math input	13 (65%)
Math editor	4 (20%)
Math language	3 (15%)

According to the user study, the majority of participants agree that our design of the user interface allows them to edit the result of HME recognition as they want. Also, from the user study, we obtained several ideas for improving the user interface.

The flowchart of the editing process is as follows.

- If the user taps on the control box, the system enables the user to modify symbol recognition.
- If the user drags the control box, the system enables the user to modify the structure relation.
- If the user draws a gesture stroke, the system handles the editing segmentation.
- Otherwise, the system discards the stroke.



## 6 Future works

The clustering methods for offline handwritten mathematical expression show that it can on the rendered online data. In practice, the real offline HME image may contain heavy noise and faint ink. Moreover, the answers in the datasets we used were just copied from given patterns so the diversity was still limited. Hence the method needs further investigations and evaluations though it showed promising results.

We have been implementing a tool for collecting handwritten patterns on the internet and starting to recruit participants. The participants are allowed to write the solutions step by step. Moreover, we will time the test to make sure the participants have pressure feeling like a real examination. The author expects that the dataset will help to evaluate the proposed clustering methods in practical circumstances.

Currently, the proposed clustering methods still need to predefine the number of clusters which makes ordinary users have difficult to use the system. Some density-based clustering techniques do not require to specify the number of clusters such as DBSCAN, OPTICS, BIRCH, etc. Though those methods' parameters also need to set manually, we can use statistical methods to set these parameters based on the input data. The methods will become flexible and even automatic.

Semantic clustering is also a considerable approach. For example, the offline HME clustering considers “0.5”, “ $1/2$ ”, “ $1 \div 2$ ” differently though semantically they are the same. To solve this problem, we may need a high reliable offline HME recognizer. However, the current performance is still below the acceptable threshold like we already said.

Also, the current model needs to be combined with a marking interface to form a complete system. Given a dataset, we can let teachers mark them first to obtain the ground truth then utilize it to evaluate our marking cost. Since our marking cost is designed to estimate the marking workload without measure the marking time directly, we can evaluate its accuracy with a real circumstance.

For the proposed interactive user interface, we also need to develop a complete system. We can use it to conduct an online examination which is flexible and easy to store. Moreover, there are some reasons in which a normal examination cannot be conducted (such as disease), our system can be adopted.

## 7 Conclusions

This thesis presented the works on applying technology to improve the quality of teaching and learning process. Particularly, we proposed an offline handwritten math expression clustering method for assisting teachers to mark mathematical assignments and examinations. Also, we presented an approach for generating handwritten math expressions artificially to create a synthetic dataset to encourage research on this topic. Besides, we designed an interactive user interface for inputting math expressions by writing on a pen-based device.

According to the experiment result, our proposed clustering method can reduce the marking workload by up to 40%. We also showed that our marking cost function can be used as a trade-off measurement between purity and number of clusters to evaluate the marker's effort. While the performance of handwritten math expressions recognizer is still limited, developing a method based on low-level features is worth considering.

The computer-assisted marking problem is a promising approach for teachers to mark descriptive answers but it is still premature. Hence, we proposed an approach for generating a public synthetic dataset of handwritten math expressions and allow other researchers to use the dataset to evaluate and compare with each other. Moreover, the method can be used to enrich the datasets for training a handwritten math expression recognizer as well.

Finally, we designed an interactive user interface for inputting math expressions to a computer easily. The user interface can be implemented on a pen-based system and allow the users to modify the recognition result while writing flexibly. In the future, this user interface can be applied in mathematical examinations which are taken place online or utilize electronic devices to input answers.

## References

- [1] L. Anthony, J. Yang and K. R. Koedinger, "A paradigm for handwriting-based intelligent tutors," *International Journal of Human-Computer Studies*, vol. 70, no. 11, pp. 866-887, 2012.
- [2] F. W. Blackwell and R. H. Anderson, "An on-line symbolic mathematics system using handprinted two-dimensional notation," *Proceedings of the 1969 24th National Conference (ACM'69)*, pp. 551-557, 1969.
- [3] H. Mouchère, C. Viard-Gaudin, R. Zanibbi, and U. Garain, "ICFHR2016 CROHME: competition on recognition of online handwritten mathematical expressions," in *Proceeding of the 15th International Conference on Frontiers in Handwriting Recognition*, Shenzhen, China, 2016.
- [4] H. Mouchère, C. Viard-Gaudin, R. Zanibbi, and U. Garain, "ICFHR 2014 competition on recognition of on-line handwritten mathematical expressions (CROHME 2014)," in *Proceeding of the 14th International Conference on Frontiers in Handwriting Recognition*, Heraklion, Greece, 2014.
- [5] 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," *International Journal on Document Analysis and Recognition*, vol. 19, no. 2, pp. 173-189, 2016.
- [6] H. Mouchère, C. Viard-Gaudin, R. Zanibbi, U. Garain, D. H. Kim, and J. H. Kim, "ICDAR 2013 CROHME: Third International Competition on Recognition of Online Handwritten Mathematical Expressions," *Proceeding of the International Conference on Document Analysis*, pp. 1428-1432, 2013.
- [7] 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," *International Journal on Document Analysis and Recognition*, vol. 19, no. 2, pp. 173-189, 2016.
- [8] H. Mouchère, C. Viard-Gaudin, R. Zanibbi, and U. Garain, "CROHME: competition on recognition of online handwritten mathematical expressions," *Proceeding 15th International Conference on Frontiers in Handwriting Recognition*, pp. 607-612, 2016.
- [9] M. Mahdavi, R. Zanibbi, H. Mouchère, C. Viard-Gaudin, and U. Garain, "ICDAR 2019 CROHME + TFD: Competition on recognition of handwritten mathematical expressions and typeset formula detection," *Proceeding 15th International Conference on Document Analysis and Recognition*, pp. 922-927, 2019.
- [10] D. H. Wang, "ICFHR 2020 Competition on Offline Recognition and Spotting of Handwritten Mathematical Expressions," *Proceeding ICFHR*, 2020.
- [11] Ronan Cummins, Meng Zhang, and Ted Briscoe, "Constrained Multi-Task Learning for Automated Essay," *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, vol. 1, pp. 789-799, 2016.
- [12] Salvatore Valenti, Francesca Neri, and Alessandro Cucchiarelli, "An overview of current research on automated essay grading," *Journal of Information Technology Education*, vol. 2, pp. 319-330, 2003.

- [13] T. Ishioka and M. Kameda, "Automated Japanese essay scoring system: Jess," *Proceedings of the 15th International Workshop on Database and Expert Systems Applications*, pp. 4-8, 2004.
- [14] Sargur Srihari, Jim Collins, Rohini Srihari, Harish Srinivasan, Shravya Shetty, Janina Brutt-Griffler, "Automatic scoring of short handwritten essays in reading comprehension tests," *Artificial Intelligence*, vol. 172, no. 2-3, pp. 300-324, 2008.
- [15] E. L. Glassman, J. Scott, R. Singh, P. J. Guo, and R. C. Miller, "Overcode: Visualizing variation in student solutions to programming problems at scale," *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 22, no. 2, pp. 129-130, 2015.
- [16] D. J. Malan, "CS50 sandbox: secure execution of untrusted code," *Proceeding of the 44th ACM technical symposium on Computer science education*, pp. 141-146, 2013.
- [17] M. T. Helmick, "Interface-based programming assignments and automatic grading of java programs," *Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education*, vol. 39, pp. 63-67, 2007.
- [18] S. Basu, C. Jacobs, and L. Vanderwende, "Powergrading: a clustering approach to amplify human effort for short answer grading," *Transactions of the Association for Computational Linguistics*, vol. 1, pp. 391-402, 2013.
- [19] M. Brooks, S. Basu, C. Jacobs, and L. Vanderwende, "Divide and correct: Using clusters to grade short answers at scale," *Proceedings of the first ACM conference on Learning at scale conference*, pp. 89-98, 2014.
- [20] D. Xu and Y. Tian, "A Comprehensive Survey of Clustering Algorithms," *Annals of Data Science*, vol. 2, no. 2, p. 165-193, 2015.
- [21] MacQueen J, "Some methods for classification and analysis of multivariate observations," *Proc Fifth Berkeley Symp Math Stat Probab*, vol. 1, p. 281-297, 1967.
- [22] G. S. a. L. G. Shapiro, *Computer Vision*, Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001.
- [23] H. Xiong, J. Wu, and J. Chen, "K-means clustering versus validation measures: A data-distribution perspective," *IEEE Transactions on Systems, Man, and Cybernetics* 39, p. 318-331, 2009.
- [24] D. Arthur and S. Vassilvitskii, "k-means++: The Advantages of Careful Seeding," in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, New Orleans, 2007.
- [25] H. S. Park and C. H. Jun, "A simple and fast algorithm for K-medoids clustering," in *Expert systems with applications*, Pohang , 2009.
- [26] L. Kaufman and P. J. Rousseeuw, "Partitioning around medoids (program pam)," in *Finding Groups in Data: An Introduction to Cluster Analysis*, Hoboken, Wiley, 1990, pp. 68-125.
- [27] L. Kaufman, P. J. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*, vol. 344, John Wiley & Sons, 2008.
- [28] R. Sibson, "SLINK: An optimally efficient algorithm for the single-link cluster method," *The Computer Journal*, vol. 16, no. 1, p. 30-34, 1973.
- [29] C. E. Rasmussen, "The Infinite Gaussian Mixture Model," in *Neural Information Processing Systems* , 1999.

- [30] X. Xu, M. Ester, H. P. Kriegel and J. Sander, "A distribution-based clustering algorithm for mining in large spatial databases," in *Data Engineering*, Orlando, 1998.
- [31] M. Ester, H. P. Kriegel, J. Sander, X. Xu, "A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, 1996.
- [32] M. Ankerst, M. M. Breunig, H. P. Kriegel and J. Sander, "OPTICS: ordering points to identify the clustering structure," in *Proceedings on 1999 AC M SIGMOD international conference on management of data*, 1999 .
- [33] S. I. Nikolenko, "Synthetic Data for Deep Learning2019," [Online]. Available: <https://arxiv.org/abs/1909.11512>. [Accessed 2020].
- [34] "http://yann.lecun.com/exdb/mnist/," [Online].
- [35] "http://graphics.cs.brown.edu/research/pcc/symbolRecognitionDataset.zip," [Online].
- [36] G. Labahn, E. Lank, M. Marzouk, A. Bunt, S. MacLean and D. Tausky, "MathBrush: a case study for pen-based interactive mathematics," in *Proceedings of the 5th Eurographics conference on Sketch-Based Interfaces and Modeling*, Annecy, France, 2008.
- [37] K.-F. Chan and D.-Y. Yeung, "Error detection, error correction and performance evaluation in on-line mathematical expression recognition," *Pattern Recognition Letters*, vol. 34, no. 8, pp. 1671-1684, 2001.
- [38] D. Zwillinger, *CRC Standard Mathematical Tables and Formulae*, CRC Press, 2011.
- [39] A.-M. Awal, H. Mouchère, and C. Viard-Gaudin, "Towards handwritten mathematical expression recognition," in *Proceeding of the International Conference on Document Analysis and Recognition*, Barcelona, Spain, 2009.
- [40] S. Quiniou, H. Mouchère, S. Saldarriaga, C. Viard-Gaudin, E. Morin, S. Petitrenaud, and S. Medjkoune, "Hamex - a handwritten and audio dataset of mathematical expressions," in *Proceeding of the 11th International Conference on Document Analysis and Recognition*, Beijing, China, 2011.
- [41] J. Stria, M. Bresler, D. Prusa, and V. Hlavc, "Mfrdb: Database of annotated on-line mathematical formulae," in *Proceeding of the 13th International Conference on Frontiers in Handwriting Recognition*, Bari, Italy, 2012.
- [42] F. D. J. Aguilar and N. S. Hirata, "Expressmatch: a system for creating ground-truthed datasets of online mathematical expressions," in *Proceeding of 10th IAPR International Workshop on Document Analysis Systems*, Queensland, Australia, 2012.
- [43] H. Mouchère, C. Viard-Gaudin, D. H. Kim, J. H. Kim, and G. Utpal, "CROHME2011: Competition on recognition of online handwritten mathematical expressions," in *Proceeding of the 11th International Conference on Document Analysis and Recognition*, Beijing, China, 2011.
- [44] H. Mouchère, C. Viard-Gaudin, D. H. Kim, J. H. Kim, and G. Utpal, "ICFHR 2012 competition on recognition of on-line mathematical expressions (CROHME 2012)," in *Proceeding of the 13rd International Conference on Frontiers in Handwriting Recognition*, Bari, Italy, 2012.

- [45] H. Mouchere, C. Viard-Gaudin, R. Zanibbi, and U. Garain, "ICFHR 2014 competition on recognition of on-line handwritten mathematical expressions (CROHME 2014)," *2014 14th International Conference on Frontiers in Handwriting Recognition*, pp. 791-796, 2014.
- [46] F. Álvaro, J.-A. Sánchez, and J.-M. Benedí, "Unbiased evaluation of handwritten mathematical expression recognition," in *Proceeding of the 13rd International Conference on Frontiers in Handwriting Recognition*, Bari, Italy, 2012.
- [47] K. Sain, A. Dasgupta, and U. Garain, "EMERS: a tree matching-based performance evaluation of mathematical expression recognition systems," *International Journal on Document Analysis and Recognition*, vol. 14, no. 1, pp. 75-85, 2011.
- [48] R. Zanibbi, A. Pillay, H. Mouchere, C. Viard-Gaudin, and D. Blostein, "Stroke-based performance metrics for handwritten mathematical expressions," in *Proceeding of the 11th International Conference on Document Analysis and Recognition*, Beijing, China, 2011.
- [49] H. Bandoh, T. Fukushima, N. Kato, and M. Nakagawa, "User Interfaces for Correcting Errors in Writing-box-free Recognition of Handwritten Text," *Transaction of IPS Japan*, vol. 43, no. 6, pp. 1996-2005, 2002.
- [50] M. Nakagawa and M. Yorifuji, "Error Correction Gestures for Free-format Text Input by Pen Interface," *Proceedings of the Human Interface Symposium*, vol. 1, pp. 43-46, 2005.
- [51] S. Smithies, K. Novins, and J. Arvo, "A handwriting-based equation editor," *Graphics Interface*, vol. 99, pp. 84-91, 1999.
- [52] E. M. Taranta II, A. N. Vargas, S. P. Compton, and J. J. Laviola Jr., "A Dynamic Pen-Based Interface for Writing and Editing Complex Mathematical Expressions With Math Boxes," *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 6, no. 2, pp. 1-25, 2016.
- [53] S. Tokuda, A. D. Le and M. Nakagawa, "Design and Prototyping of Error Correction Interface for Inputting Mathematical Expressions by Handwriting Recognition," *IPSJ SIG Technical Report*, Vols. 2016-CE-133, no. 8, pp. 1-6, 2016.
- [54] T. V. Phan, J. Gao, B. Zhu, and M. Nakagawa, "Effects of Line Densities on Nonlinear Normalization for Online Handwritten Japanese Character Recognition," in *Document Analysis and Recognition (ICDAR)*, Beijing, 2011.
- [55] C. L. Liu, K. Marukawa, "Pseudo two-dimensional shape normalization methods for handwritten Chinese character recognition," *Pattern Recognition*, vol. 38, no. 12, p. 2242-2255, 2005.
- [56] C. L. Liu, and Katsumi Marukawa, "Pseudo two-dimensional shape normalization methods for handwritten Chinese character recognition," *Pattern Recognition*, vol. 38, no. 12, pp. 2242-2255, 2005.
- [57] C. L. Liu, Ying-Jian Liu, Ru-Wei Dai, A. C. Downton, and S. Impedovo, "Preprocessing and statistical/structural feature extraction for handwritten numeral recognition," *Progress of Handwriting Recognition*, pp. 161-168, 1997.
- [58] D. J. C, "Introduction to Statistical Pattern Recognition," *Computers and Geosciences*, vol. 7, no. 22, pp. 833-834, 1996.
- [59] Ramadhan I., B. Purnama, and S. Al Faraby, "Convolutional neural networks applied to handwritten mathematical symbols classification," *Proceeding 2016 4th International Conference on Information and Communication Technology (ICoICT)*, pp. 1-4, 2016.

- [60] D. H. Hai, A. D. Le, and M. Nakagawa, "Deep neural networks for recognizing online handwritten mathematical symbols," *Proceeding 2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, pp. 121-125, 2015.
- [61] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," *International conference on machine learning*, pp. 478-487, 2016.
- [62] M. Caron, B. Piotr, J. Armand, and M. Douze, "Deep clustering for unsupervised learning of visual features," *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 132-149, 2018.
- [63] S. J. Rao, Y. Wang, and G. W. Cottrell, "A Deep Siamese Neural Network Learns the Human-Perceived Similarity Structure of Facial Expressions Without Explicit Categories," *CogSci.*, 2016.
- [64] J. Chang, L. Wang, G. Meng, S. Xiang, and C. Pan, "Deep adaptive image clustering," *Proceedings of the IEEE international conference on computer vision*, pp. 5879-5887, 2017.
- [65] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science* 313, vol. 313, no. 5786, pp. 504-507, 2006.
- [66] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua, "A convolutional neural network cascade for face detection," *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5325-5334, 2015.
- [67] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Is object localization for free?-weakly-supervised learning with convolutional neural networks," *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 685-694, 2015.
- [68] D. Yoo, S. Park, J. Y. Lee, and I. S. Kweon, "Multi-scale pyramid pooling for deep convolutional representation," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 71-80, 2015.
- [69] S. Ren, K. He, R. Girshick, J. Sun, "Faster RCNN Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2017.
- [70] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431-3440, 2015.
- [71] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2921-2929, 2016.
- [72] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 9, pp. 1904-1916, 2015.
- [73] Khuong V. T. M., Huy U. Q., Nakagawa M., and Phan M. K., "Generating Synthetic Handwritten Mathematical Expressions from a LaTeX Sequence or a MathML Script," *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pp. 922-927, 2019.
- [74] I. King M.-C. Yuen, and K.-S. Leung, "A survey of crowdsourcing systems," in *Proceeding of the IEEE 3rd International Conference on Privacy, Security, Risk and Trust and IEEE 3rd International Conference on Social Computing*, Boston, USA, 2011.

- [75] T. S. F. Haines, Oisín M. Aodha, and G. J. Brostow, "My text in your handwriting," *ACM Transactions on Graphics*, vol. 35, no. 3, pp. 26:1-26:18, 2016.
- [76] N. Journet, M. Visani, B. Mansencal, K. V. Cuong, and A. Billy, "Doccreator: A new software for creating synthetic ground-truthed document images," *Journal of imaging*, vol. 3, no. 4, pp. 62:1-62:17, 2017.
- [77] P. Krishnan and C. V. Jawahar, "Generating synthetic data for text recognition," arXiv:1608.04224, 2016.
- [78] A. Graves, "Generating sequences with recurrent neural networks," arXiv:1308.0850, 2013.
- [79] U. Bhattacharya, R. Plamondon, S. D. Chowdhury, P. Goyal, and S. K. Parui, "A sigma-lognormal model-based approach to generating large synthetic online handwriting sample databases," *International Journal on Document Analysis and Recognition*, vol. 20, no. 3, pp. 155-171, 2017.
- [80] K. Davila, S. Ludi, and R. Zanibbi, "Using Off-Line Features and Synthetic Data for On-Line Handwritten Math Symbol Recognition," in *Proceeding of the 14th International Conference on Frontiers in Handwriting Recognition*, Heraklion, Greece, 2014.
- [81] A. M. Awal, H. Mouchère, and C. Viard-Gaudin, "Towards handwritten mathematical expression recognition," in *Proceeding of the 10th International Conference on Document Analysis and Recognition*, Barcelona, Spain, 2009.
- [82] A. D. Le and M. Nakagawa, "Training an End-to-End System for Handwritten Mathematical Expression Recognition by Generated Patterns," in *Proceeding of the 14th IAPR International Conference on Document Analysis and Recognition*, Kyoto, Japan, 2017.
- [83] A. D. Le and M. Nakagawa, "A system for recognizing online handwritten mathematical expressions by using improved structural analysis," *International Journal on Document Analysis and Recognition*, vol. 19, no. 4, pp. 305-319, 2016.



## AUTHOR'S PUBLICATIONS

- **Journal papers**

**Vu Tran Minh Khuong**, Minh Khanh Phan, Huy Quang Ung, Cuong Tuan Nguyen, and Masaki Nakagawa, "Clustering of Handwritten Mathematical Expressions for Computer-Assisted Marking," *IEICE Trans. Information and Systems*, Vol. E104-D, No. 2, pp. 275-284, 2021 (Chapter 3).

Cuong Tuan Nguyen, **Vu Tran Minh Khuong**, Hung Tuan Nguyen, and Masaki Nakagawa, "CNN based spatial classification features for clustering offline handwritten mathematical expressions," *Pattern Recognition Letters*, Vol. 131, pp. 113-120, 2020 (Chapter 3).

- **International conferences**

**Vu Tran Minh Khuong**, Huy Quang Ung, Masaki Nakagawa, and Minh Khanh Phan, "Generating Synthetic Handwritten Mathematical Expressions from a LaTeX Sequence or a MathML Script," *Proceedings of 15th International Conference on Document Analysis and Recognition (ICDAR)*, Sydney, Australia, pp. 922-927, 2019 (Chapter 4).

**Vu Tran Minh Khuong**, Minh Khanh Phan, and Masaki Nakagawa, "Interactive Interfaces for Recognizing Online Handwritten Mathematical Expressions and Correcting Misrecognitions," *Proceedings of 2nd International Workshop on Open Services and Tools for Document Analysis (ICDAR-OST)*, Sydney, Australia, pp. 26-30, 2019 (Chapter 5).

**Vu Tran Minh Khuong**, Huy Quang Ung, Cuong Tuan Nguyen, Masaki Nakagawa, "Clustering Offline Handwritten Mathematical Answers for Computer-Assisted Marking," International Conference on Pattern Recognition and Artificial Intelligence, May 14-17, 2018 (Chapter 3).

- **Joint works**

Huy Quang Ung, **Vu Tran Minh Khuong**, Anh Duc Le, Cuong Tuan Nguyen, Masaki Nakagawa, "Bag-of-features for clustering online handwritten mathematical expressions," International Conference on Pattern Recognition and Artificial Intelligence, May 14-17, 2018.

Nguyen Thi Hong Nhung, **Vu Tran Minh Khuong**, Vu Quang Huy, Pham The Bao, Classifying prostate cancer patients based on total prostate-specific antigen and free prostate-specific antigen features by support vector machine. *Journal of Cancer Research and Therapeutics*, Vol. 12, pp. 818-825, 2016.