ITC 1/52
Information Technology and Control
Vol. 52 / No. 1 / 2023
pp. 169-184
DOI 10.5755/j01.itc.52.1.32066

A Step Toward an Automatic Handwritten Homework Grading System for Mathematics

Received 2022/08/12 | Accepted after revision 2023/01/30

https://doi.org/10.5755/j01.itc.52.1.32066

HOW TO CITE: Chaowicharat, E., Dejdumrong, N. (2023). A Step Toward an Automatic Handwritten Homework Grading System for Mathematics. *Information Technology and Control*, 52(1), 169-184. https://doi.org/10.5755/j01.itc.52.1.32066

A Step Toward an Automatic Handwritten Homework Grading System for Mathematics

Ekawat Chaowicharat
Department of Mathematics, Faculty of Science, Mahidol University, Thailand; phone: +66-95-9670063; e-mail: ekawat.cha@mahidol.ac.th

Natasha Dejdumrong
Department of Computer Engineering, Faculty of Engineering, King Mongkut's University of Technology Thonburi, Thailand; phone: +66-85-4545252; e-mail: natasha.dej@mail.kmutt.ac.th

Corresponding author: ekawat.cha@mahidol.ac.th

An automatic system that helps teachers and students verify the correctness of handwritten derivation in mathematics homework is proposed. The system acquires input image containing handwritten mathematical derivation. In our preliminary study, the system that comprises only mathematical expression recognition (MER) and computer algebra system (CAS) did not perform well due to high misrecognition rate. Therefore, our study focuses on fixing the misrecognized symbols by using symbols replacement and the surrounding information. If all the original mathematical expressions (MEs) in the derivation sequence are already equivalent, the derivation is marked as "correct". Otherwise, the symbols with low recognition confidence will be replaced by other possible candidates to maximize the number of equivalent MEs in that derivation. If there is none of symbols replacement that makes every line equivalent, the derivation is marked as "incorrect". The recursive expression tree comparison was applied to report the types of mistake for those problems marked as incorrect. Finally, the performance of the system was evaluated by the digitally generated dataset of 6,000 handwritten mathematical derivations. The results showed that the symbols replacement improve the F1-score of derivation step marking from 69.41 to 95.95 % for the addition/ subtraction dataset and from 61.45 to 89.95 % for the multiplication dataset when compared to the case of using raw recognized string without symbols replacement.

KEYWORDS: Mathematical expression recognition, automatic homework grading, symbols replacement.

1. Introduction

Formative assessment is best accomplished if teachers are able to quickly mark the student works and immediately return the graded results to the students. Immediate feedback can be succeeded without too much attempt when using multiple choice questions or short answer questions where an exact keyword is prepared. Most of learning management systems (LMS) nowadays has automatic grading feature for multiple choice questions (MCQ) and short answer question built-in.

However, handwritten homework assignments that show steps of derivation still plays an important role in mathematics education. It is one of the best ways to express step of thought, including logical thinking, theorem applying, and accurate calculation skill. Besides, working with a scratch paper and pencil (or electronic tablet with stylus) is still an intuitive way to solve mathematics problems [1].

Nowadays, automatic grading of handwritten homework containing the entire calculation steps by computer is not at the state of practical use due to the complication of processes, unlike that of MCQ and short answer grading, which has been widely used for a long time. The idea of automatic homework grading system has been around for decades alongside with optical character recognition development. Started with grading handwritten short answers, the most feasible method requires special marks such as printed underlining or boxes to locate the image region that contains the solution. Then the mathematical expression recognition is utilized [6, 14]. Recent works have focused on locating mathematical expressions embedded in handwritten text automatically [8, 19].

The more challenging task is the entire derivation grading. Since the different pathways of derivation could lead to the same correct answer, while a tentative guideline for mathematics homework checking could be utilized, not every correct derivation is written in the exact same way. The grading system has to be flexible enough to accept every correct pathway to the final conclusion.

In this study, we propose an automatic system that performs handwritten mathematics homework grading from images containing all the calculation steps. The core idea is to combine a mathematical expression recognition (MER) and a computer algebra system (CAS). MER converts the mathematical expression (ME) from images into a sequence of symbols, where CAS then parses the expressions into expression trees and compares whether each pair of MEs from the consecutive lines are equivalent or not. If it is the case, the entire derivation is marked as correct. Conversely, if there exists ME that is not equivalent to the surroundings, that step is marked as incorrect.

Our preliminary design using convolutional neural network (CNN) OCR for symbol recognition and Sympy library in python (as a CAS) reveals a number of challenging issues in this study as follows:

- Since mathematical expressions usually contains a long sequence of symbols, the accuracy of the marking highly depends on the quality of symbol recognition. An incorrect recognized symbol could ruin the marking result of the entire derivation.
- The incorrect symbols can be caused by both the mathematical mistake from the original image, or the MER misrecognition itself. The locations of the incorrect symbols are also unable to determine when the ground truth is unknown.
- In text OCR, misrecognition can be resolved by using character sequence patterns such as the n-gram model. Yet symbol ambiguity in MER sometimes cannot be handled because of the less strict symbol sequence patterns, for instance both “x+b” and “x+6” are valid MEs. It is possible that an expression is syntactically correct, but not consistent to the surroundings. For example, 16x + 11 seems to be a valid ME, but it might be recognized from the actual expression (6x + 1) where both “(” and “)” are perceived as 1.

Therefore, the main focus of this study is to find an appropriate symbols replacement algorithm so that the effect of misrecognition is neutralized.

In this study, we designed an algorithm aiming to solve the misrecognition by using the surrounding MEs in the same derivation sequence to confirm the correctness of the symbols replacement. Our key concepts in this algorithm design consist of the following components:

- The candidates for the symbols replacement comes from the confusion matrix and the lower rank recognition candidates.

- Instead of using the language model of symbol sequence alone, we use the information from the surrounding lines in the sequence of MEs to confirm the most likely symbols replacement. For example, if x + b makes that step of calculation consistent with the next line, it is more likely to be correct rather than x + 6.
- With the unavailability of mathematical derivation dataset, the derivation images will be generated from the dataset of single mathematical symbols. The generated dataset contains some intentional calculation mistake to test whether the system can detect the mistake from the source image or not.

2. Literature Review

The literature review comprises five parts, including automatic homework marking, mathematical expression recognition, mathematical expression simplification, confusion matrix and error correction, and common mistakes in algebraic simplification.

2.1. Automatic Homework Marking (AHM)

Automatic homework marking has been proposed and used for a long time. Due to many advantages, such as, saving of manpower and time, better consistency of grading criteria, bias free, and instantaneous feedback [24].

The development path of AHM started from automatic mark recognition that has been used in MCQ test [21]. This method uses fixed format answer sheet. Answers are chosen by filling the circle with black pencil. It can also be applied to the digits answers by using an array of circles.

Short answer marking works with both text and numeric answers. Localization and recognition are used for detecting and converting the handwritten answers into digital format. It needs text box or underline as the marker, so that pattern recognition be able to locate and recognize text in that area [14].

Long answer/ free-text marking is mostly used with digital text input since it is not practical for handwriting due to the limitation of character recognition technology. For free-text input, the system has to analyze sentence structure and extract keywords that match the key answer [25]. Another field of practical use is the automatic code assessment, where the system can analyze the source code in a programming language and return the evaluation in terms of syntax, plagiarism, semantic, performance, or quality [4, 5].

There are disadvantages of AHM when compared to human marking. First, the marking criteria must be simple enough. Submission that does not meet the criteria will be rejected since it is hard to design the machine to be as flexible as human grader. As a consequence, the answer sheets for AHM are in the less flexible format and also limit the type of questions to assign. The limited type of the test and assignment that computer can grade also limit the creativity of teachers to design tests, and also limit the cognitive level required to answer the question [9].

From a study, the paper-and-pencil test (PPT) still have slightly better learning outcome than using computer-based system because PPT allows inspection of high cognitive demand tasks [23]. In this study, we are looking for a solution to bridge the gap between the automatic marking by computer and the handwritten, free text homework, especially in mathematics.

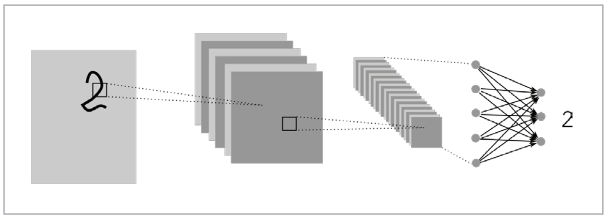
2.2. Mathematical Expression Recognition (MER)

Mathematical expression recognition (MER) is the software that converts mathematical expression (ME) images into a machine-understandable format [3]. The purpose of MER is to offer an alternative ME input method that is seamless to human writing, without the use of markup languages or any point-and-click interface. MER is a special case of optical character recognition (OCR) with the domain restricted to mathematical expressions and symbols. MER consists of the following common procedures.

Math expression localization and symbol segmentation: ME is typically written in line with text and must be extracted before recognition. A number of techniques can be used to distinguish ME from plain text, for example, global features that take aspect ratio, height, deviation of coordinates from the main line, and support vector machine classifier [7]. Some systems use a modern approach such as U-Net to separate ME regions from text [19].

Symbol recognition: The convolutional neural network (CNN) is a widely used deep learning algorithm that transforms input images into symbol classes directly without any predefined features. The earlier stages of CNN consist of the convolution and max pooling process which are used to learn the pixel patterns into feature vectors. The learned features are then passed through the feedforward network to perform the classification, as shown in Figure 1. A number of MERs have been developed recently based on CNN [14, 16, 17, 20].

Figure 1
Structure of CNN



Structural analysis: Structural analysis is the arrangement of recognized symbols into the expression with a valid syntax and then involves semantic extraction. Structural analysis can be as simple as the rule-based methods that use spatial relation. For example, the upper and lower bounds of the symbols can be used to classify the superscript from characters in the main line [10]. Machine learning algorithms such as neural network and SVM can also be applied to classify the spatial relation between a group of symbols into the mathematical syntax [22].

The state of the art for MER is based on encoder-decoder network, where the encoder transforms the entire image of ME into a sequence of vector representation, then the decoder converts the representation into a 1D latex syntax, or 2D expression tree structure. In the present, expression recognition rates (ExpRate) of encoder-decoder network are around 65% [27, 28, 29]. One of the limitations is the lack of large public dataset for the handwritten ME [1].

2.3. Mathematical Expression Simplification

Simplification is the process of rewriting ME with minimal numbers of items and variables. The process transforms ME into an equivalent ME which reduces complication. Most of simplification process is based on representing ME as an abstract syntax tree (AST). Then apply the term rewrite rules by traversing the AST to match the subtree that is consistent with the rule for transformation [11].

2.4. Confusion Matrix and Error Correction

The confusion matrix of character recognition contains information about the misrecognized symbol which is used for both system evaluation and error correction. Since the internal redundancy of symbols sequence is measured by a probabilistic language model such as N-gram, when symbol sequences with low consistency are found, other candidates in the confusion matrix that provide better consistency can be utilized [12, 15].

The problem of using error correction in the mathematical context is that the choice of symbols in ME rather depends on the surrounding MEs in the sequence than the surrounding symbols in the same ME itself. Therefore, the language model is not appropriate for error correction in our problem, but the confusion matrix still can be applied in some other ways.

2.5. Common Mistakes in Algebraic Simplification

Common mistakes are the miscalculations usually found in mathematics newcomers, comes from the lack of understanding, or the careless during the calculation, such as doing addition in place of multiplication.

Some common mistakes in algebraic simplification are presented in Table 1. We then apply some types of mistakes to generate the derivation images containing intentional error and use these derivation images for our system evaluation.

Table 1
Types and examples of mathematical derivation mistake

(1) Miscalculation (add and multiply)	$A \times B = A + B$	$3 \times 2 = 5$
(2) Ignorance of variable	$A + BX = A + B$	$3 + 5x = 8$
(3) Add or multiply exponents	$X^A X^B = X^{AB}$	$x^3 x^2 = x^6$
(4) Minus sign with square	$-A^2 = A^2$	$-5^2 = 25$
(5) Improper distribution	$A(B + C) = AB + C$	$3(x + 2) = 3x + 2$
(6) Improper cancellation	$\frac{AB + C}{A} = B + C$	$\frac{8 - 5}{2} = 4 - 5$
(7) Additive assumption	$(X + Y)^A = X^A + Y^A$	$(x + 3)^2 = x^2 + 9$

In conclusion, the current technology seems to be feasible for creating a handwritten mathematical homework marking system. However, using the information from the confusion matrix alone cannot guarantee the practical result since there can be multiple candidates that make valid MEs. Therefore, we aim at using the context from the other MEs in the same derivation to determine the most likely symbol replacement.

It is also possible that the surrounding MEs might contain some error as well, so the replacement algorithm is not just fixing each line and compare to the reliable reference, but every line in the derivation needs to be processed at the same time.

To narrow the scope of this study, we simulated only two types of mistake from Table 1, including the miscalculation of addition and multiplication (1) and addition or multiplication of exponent (3) with the problem of polynomial simplification in our dataset generator.

3. System Architecture

3.1. System Structure

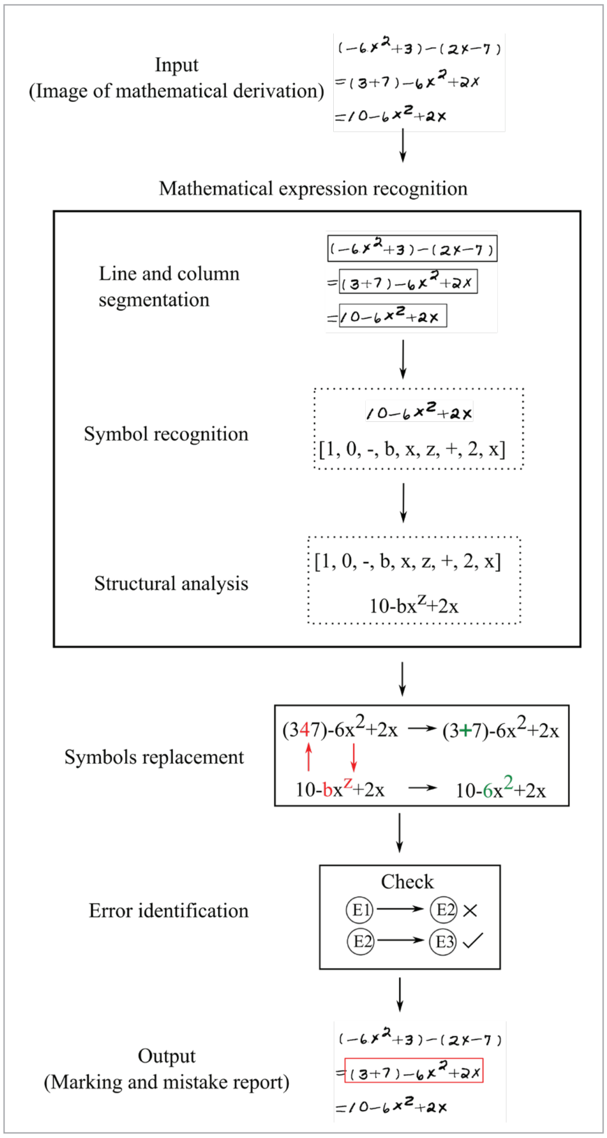
The designed system acquires an image of a student's homework that contains a sequence of handwritten mathematical expressions written in each line. The system then analyses and returns the marking output indicating whether each derivation lines is correct or not, together with short comments on the type of mistake (if any). The system components contain 3 modules as shown in Figure 2.

1 Mathematical expression recognition: includes line and column segmentation, mathematical symbol recognition, and structural analysis.

2 Symbols replacement: uses the confusion matrix from symbol recognition and the fast ME matching to find an appropriate symbols replacement that make the ME consistent with the surrounding MEs.

3 Mathematical error identification: comprises a computer algebra system and an expression tree comparison. This module generates report that identifies the mismatches for the inconsistent derivation.

Figure 2
Workflow of the overall system structure

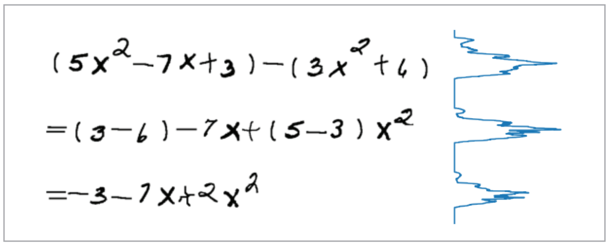


3.2 Mathematical Expression Recognition

3.2.1. Line Segmentation

The input image containing multiple lines of MEs can be segmented by using the histogram projection into sub-images that contains only one expression each. Let M be a 2D array representing the inverted input image (black background and white foreground) with dimension $m \times n$. The projection $P(M, y) = \sum_{i=1}^{n} M_{y,i}$ represents the sum of pixel values along the y^{th} row. In the binarized image, the interval of y where $P(M, y) = 0$ is marked as space between lines, as shown in Figure 3.

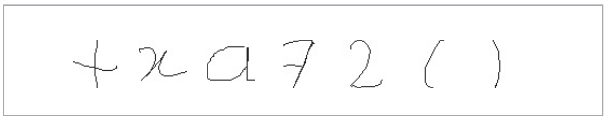
Figure 3
Pixel histogram shows image with 3 lines and its projection



3.2.2. Mathematical Symbol Recognition

To extract foreground images of symbols from the background, we used the Connected-Component command from opencv library with the default 8-way connectivity configuration. Each of the connected components is recognized by a convolutional neural network (CNN) trained by the dataset of mathematical symbols from CROHME and MNIST. The original dataset contains 82 classes. However, only 32 classes consisting of digits, binary operators, parenthesis, and variables are included in this study. Some examples of symbol images are shown in Figure 4.

Figure 4
Sample images of symbols +, x, a, 7, 2, (, and) from the dataset



The training set is augmented by the modified images using dilation with the circular kernel of radius ranging from 1 to 5 pixels and rotation ranging from -10 to 10 degree to prevent the overfitting.

We used the CNN library from TensorFlow. The detailed structure of the CNN is as follows:

- Input size 45×45 pixels
- Convolutional layer with 128 features, kernel size 3 × 3, ReLU activation function
- Max pooling with pool size 2 × 2
- Another convolutional layer with 128 features, kernel size 3 × 3, ReLU activation function
- Dropout rate 0.25
- Flatten layer
- Dense layer with 256 nodes, ReLU activation function.
- Output layer with 32 nodes (equal to the number of symbols), softmax activation function

The first few highest score (up to 5) candidates from CNN are selected. The list of candidates together with the coordinates are brought to the structural analysis.

3.2.3. Structural Analysis

Since the scope of this study is just polynomial simplification, the structural analysis needs to determine the main line and superscript only. Symbols with the lowest pixel located above the center of the main line will be counted as a superscript. The more complicated structural analysis can be used when other types of mathematical expressions are included in the future.

3.3. Symbols Replacement

As mentioned in the introduction, the inconsistent MEs between lines come from MER misrecognition or the calculation mistake written in the source image. If the ME from the image is incorrect at the first place, the MER error will not affect the marking result because the derivation is incorrect anyway. On the other hand, if the ME from the image is correct, the MER error can cause the false negative in the marking result. The aim of this process is to find the symbols replacement where the MER output is misrecognized in order to make the longest sequence of equivalent MEs possible.

We first setup the notation for the symbols replacement as follows:

Mathematical symbol (s_i^l): the i^{th} symbol from line l, including numbers, mathematical constant, variables, operators, and brackets.

Raw string from line l (R^l): consists of symbols $s_1^l, s_2^l, s_3^l, \ldots, s_n^l$ where n is the length of the string. R^l is not necessary to be a valid mathematical expression since the misrecognized symbols can be fixed later.

Replacement mapping ($M^l : s_i^l \rightarrow s_i^{l*}$): mapping between the old symbols and their replacement at each position. The depth of replacement (d) in line l is the number of index i where $s_i^l \neq s_i^{l*}$. The mapping for every line altogether is denoted by $M = (M^1, M^2, \ldots, M^L)$.

Valid expression (E^l), the string of symbols from line l that is parsable by the context free grammar of the standard mathematical expression. E^l can be converted into a corresponding expression tree T^l.

Modified string ($M^l(R_i^l)$) is the result of applying a replacement M^l to the raw string R^l. If $M^l(R_i^l)$ is parsable, then $E^l = M^l(R_i^l)$. Otherwise, E^l does not exist.

Mathematical equivalent: a pair of two distinct valid expressions E^{l1} and E^{l2} is said to be mathematical equivalent if the difference between two of them can be simplified to zero. The equivalent expressions always return the same output when we plug in the same value(s) of the variable(s).

Let $K(M,R)$ be the set of equivalent MEs after replacing the sequence of raw string $R = [R^1, R^2, ..., R^m]$ by the mapping M. The purpose is to find the mapping M that maximizes the number of equivalent MEs after the symbols replacement, i.e., $argmax_M|K(M,R)|$.

Alternative symbols in the replacement mapping comes from the candidate from the frequently misrecognized symbols in the confusion matrix (as some examples are shown in Table 2) and the lower rank recognition candidates from MER for that particular symbol. For each line of raw string, there can be more than one replacement that generate valid expressions. Therefore, checking the consistency of MEs throughout all combination of valid expression between lines is inevitable. The next goal is to make this process as fast as possible.

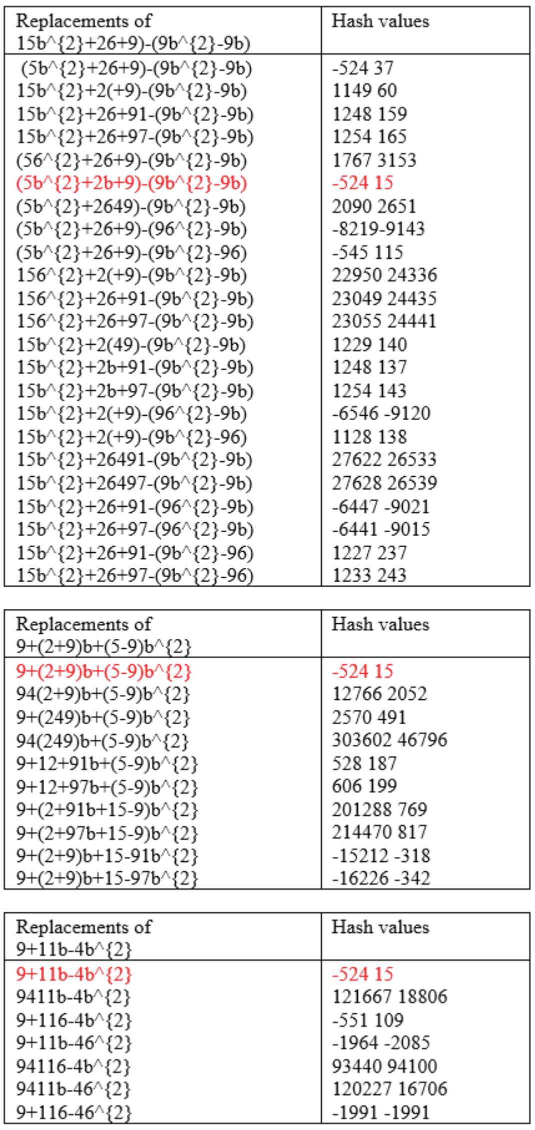
Table 2
Frequently misrecognized symbol pairs from the confusion matrix

Symbol	Misrecognized as	Symbol	Misrecognized as
a	2	4	a, y, +
d	b	9	a
z	2	+	4
2	z, a	c	(
7	)	y	x, 4
)	1, 7	1	), (, 7
b	6	6	b
x	y	(	1, c
0	a	,	), 1

3.4. Fast Pruning and Exact MEs Sequence Matching

Comparison of MEs requires multiple loops for checking whether the expression tree simplification $simp(E^{l1} - E^{l2})$ equals 0 or not, which is computationally intensive. Suppose there are m lines of derivation and $p_1, p_2, ... , p_m$ be the number of valid replacement mappings that make valid MEs, the total number of pairwise expression tree comparison trials are

Figure 5
Hashing and candidate selection



Replacements of 15b^{2}+26+9)-(9b^{2}-9b)	Hash values
(5b^{2}+26+9)-(9b^{2}-9b)	-524 37
15b^{2}+2(+9)-(9b^{2}-9b)	1149 60
15b^{2}+26+91-(9b^{2}-9b)	1248 159
15b^{2}+26+97-(9b^{2}-9b)	1254 165
(56^{2}+26+9)-(9b^{2}-9b)	1767 3153
(5b^{2}+2b+9)-(9b^{2}-9b)	-524 15
(5b^{2}+2649)-(9b^{2}-9b)	2090 2651
(5b^{2}+26+9)-(96^{2}-9b)	-8219-9143
(5b^{2}+26+9)-(9b^{2}-96)	-545 115
156^{2}+2(+9)-(9b^{2}-9b)	22950 24336
156^{2}+26+91-(9b^{2}-9b)	23049 24435
156^{2}+26+97-(9b^{2}-9b)	23055 24441
15b^{2}+2(49)-(9b^{2}-9b)	1229 140
15b^{2}+2b+91-(9b^{2}-9b)	1248 137
15b^{2}+2b+97-(9b^{2}-9b)	1254 143
15b^{2}+2(+9)-(96^{2}-9b)	-6546 -9120
15b^{2}+2(+9)-(9b^{2}-96)	1128 138
15b^{2}+26491-(9b^{2}-9b)	27622 26533
15b^{2}+26497-(9b^{2}-9b)	27628 26539
15b^{2}+26+91-(96^{2}-9b)	-6447 -9021
15b^{2}+26+97-(96^{2}-9b)	-6441 -9015
15b^{2}+26+91-(9b^{2}-96)	1227 237
15b^{2}+26+97-(9b^{2}-96)	1233 243

Replacements of 9+(2+9)b+(5-9)b^{2}	Hash values
9+(2+9)b+(5-9)b^{2}	-524 15
94(2+9)b+(5-9)b^{2}	12766 2052
9+(249)b+(5-9)b^{2}	2570 491
94(249)b+(5-9)b^{2}	303602 46796
9+12+91b+(5-9)b^{2}	528 187
9+12+97b+(5-9)b^{2}	606 199
9+(2+91b+15-9)b^{2}	201288 769
9+(2+97b+15-9)b^{2}	214470 817
9+(2+9)b+15-91b^{2}	-15212 -318
9+(2+9)b+15-97b^{2}	-16226 -342

Replacements of 9+11b-4b^{2}	Hash values
9+11b-4b^{2}	-524 15
9411b-4b^{2}	121667 18806
9+116-4b^{2}	-551 109
9+11b-46^{2}	-1964 -2085
94116-4b^{2}	93440 94100
9411b-46^{2}	120227 16706
9+116-46^{2}	-1991 -1991

$\sum_{i=1}^{m-1} p_i p_{+1}$, which is impractical in terms of CPU running time.

Instead of using pairwise expression comparison, it is much more efficient to prune out the unusable MEs before applying the simplification of expression trees. We use the fact that the equivalent expressions have the same output value when we plug in the same values of the variables.

We substitute 2 randomized constants per variable and use the numerical output as the hash values representing the ME. With the hashing, the potential equivalent expressions between lines will be chosen without too much effort by finding the identical hash values between lines. Only a few pairs of expression need to be verified by the expression tree simplification. This method takes $\sum_{i=1}^{m} p_i$ loops of evaluating the hashes, plus a much less intensive processes of the expression tree comparison. Figure 5 shows an example of finding hash values of all possible MEs among all 3 lines. The only identical hash values are (-524, 15), so there is only one candidate from each line to be verified.

3.5. Steps for the MEs Sequence Matching

It is not always the case that exact match throughout the derivation can be found from the hashing process. The following steps guarantee to return the longest equivalent MEs after the symbols replacement. The flowchart version of this algorithm can also be found in Figure 6.

Step1: Obtain the first candidate from each line. Parse string in each line and check the equivalence between any two consecutive lines.

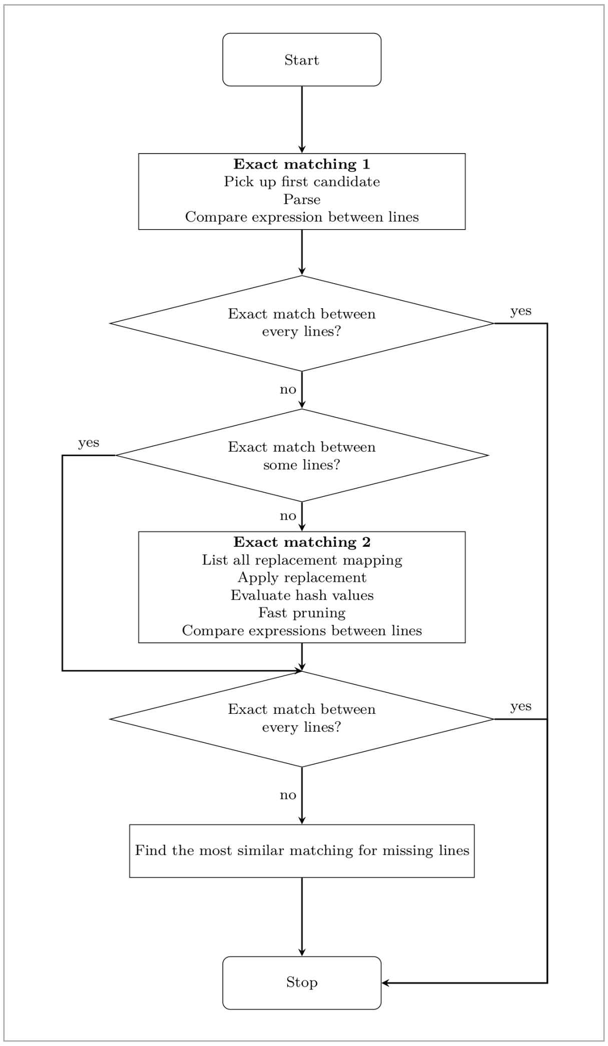
- If they are all equivalent, i.e., $R^1, \ldots, R^m$ can be parsed into valid expressions $E^1, \ldots, E^m$ where $E^1 = E^2 = E^3 = \ldots = E^m$, the matching process is satisfied and terminated here. This is the ideal case where symbols replacement is not necessary.
- If only some but not all expressions are equivalent, put them in the confident list. Then skip to the last step.
- If none of the expressions are equal, go to step 2.

Step 2: None of the raw strings are parsable or equivalent to the surrounding lines at all,

- Use all possible candidates from the confusion matrix and lower rank recognition candidates to generate the replacement map for each line of raw string, started from depth = 1 and increasing until reaching the maximum depth.
- Find the longest equivalent expression that starts from the first line (or closest to the first line if there are more than one sequence with the same length). Use the hashing mentioned earlier for the fast pruning of non-equivalent pairs, then simplify the cancellation of expressions only for a few remaining pairs.
- If there are some replacements that generate the exact match throughout every lines of derivation, return the modified MEs immediately. Otherwise, keep increasing the depth and repeat the symbols

Figure 6

Flowchart for MEs sequence matching

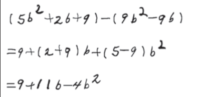

Start
Exact matching 1
Pick up first candidate
Parse
Compare expression between lines
Exact match between every lines?
yes
no
Exact match between some lines?
yes
no
Exact matching 2
List all replacement mapping
Apply replacement
Evaluate hash values
Fast pruning
Compare expressions between lines
Exact match between every lines?
yes
no
Find the most similar matching for missing lines
Stop

replacement until reaching the maximum value. If there are some raw strings that do not match any other line, put all the exact match in the confident list.
- Go to step 3.

Step 3: Fill the missing lines in the confident list with the most likely MEs.
- If the confident list is still empty in this step, choose a valid expression generated from a replacement mapping for the first line.
- Fill the missing lines by the valid ME (generated from the replacement mapping) that is similar to the previous confident expression the most. Use the expression tree comparison in the next section to evaluate the similarity.

Figure 7 (left) shows an example of an input image, together with some incorrect MER outputs (middle). After the MEs matching, the longest equivalent ME sequence can be obtained, as shown in Figure 7 (right). The expression $15b^2 + 26 + 9$ was modified into $(5b^2 + 2b + 9)$ in order to be consistent to the surrounding expressions.

Figure 7
Derivation image (left), with faulty expression from recognition output (middle), and the corrected expression from symbol substitution (right)

	15b^{2}+26+9)-(9b^{2}-9b)	(5b^{2}+2b+9)-(9b^{2}-9b)
	9+(2+9)b+(5-9)b^{2}	9+(2+9)b+(5-9)b^{2}
	9+11b-4b^{2}	9+11b-4b^{2}

3.6. Expression Tree Comparison

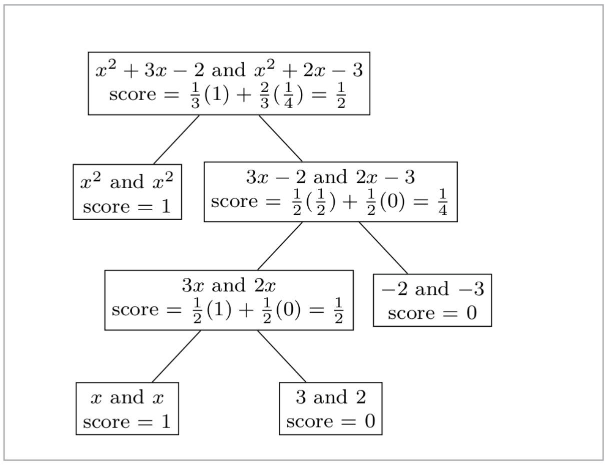
Expression tree comparison is the process to measure the similarity score between two simplified trees. A pair of expression trees with similar terms will get the similarity closer to 1 than the pair of trees that contains a lot of distinctions.

Let T^i and T^j be the expression trees of the expression E^i and E^j from line i and j of the derivation, respectively. To find out the different terms between two MEs, terms in the MEs that have been matched correctly (simplified difference returns zero) are removed from both trees. The algorithm loops over C_r^i and C_s^j, which are children nodes of the T^i.*root* and T^j.*root*, respectively. If there are subtrees T_r^i and T_s^j, rooted at C_r^i and C_s^j, such that the simplification of T_r^i and T_s^j are identical, both T_r^i and T_s^j are removed. The algorithm recurs until there is none of identical subtree left in T^i and T^j.

The similarity calculation is based on the intuition that similar subexpressions should contain similar sets of terms, functions, or constant.

Figure 8 shows an example of the tree comparison and similarity score calculation. While the cancellation was traced downward from the entire ME to each component, the similarity calculation started from the smallest term and are accumulated from bottom up. The mismatches were shown at the leaf nodes where score = 0 and will be reported in the output. Moreover, the similarity scores can be used for selecting the most likely replacement mapping for the raw string that has no exact match with any other line as well.

Figure 8
Tree comparison between $x^2 + 3x - 2$ and $x^2 + 2x - 3$


$x^2 + 3x - 2$ and $x^2 + 2x - 3$
score $= \frac{1}{3}(1) + \frac{2}{3}(\frac{1}{4}) = \frac{1}{2}$
x^2 and x^2
score = 1
$3x - 2$ and $2x - 3$
score $= \frac{1}{2}(\frac{1}{2}) + \frac{1}{2}(0) = \frac{1}{4}$
$3x$ and $2x$
score $= \frac{1}{2}(1) + \frac{1}{2}(0) = \frac{1}{2}$
-2 and -3
score = 0
x and x
score = 1
3 and 2
score = 0

4. Results

4.1. Derivation Image Generator

Since there is none of available mathematical derivation dataset labelled with the type of mistakes, we created our own dataset for this study.

Similar to the pattern generation that help improve the performance of MER [13], the derivation images were generated by arranging the separated symbol images obtained from https://www.kaggle.com/atasets/xainano/handwrittenmathsymbols that was parsed, extracted and modified from CROHME dataset, plus digits images from MNIST datasets so that we can manipulate the mistake and the correctness for each line in the derivation as desired. Among 300,000 symbol images, around 20% are used for derivation dataset generator, while the remaining 80% are reserved for training MER.

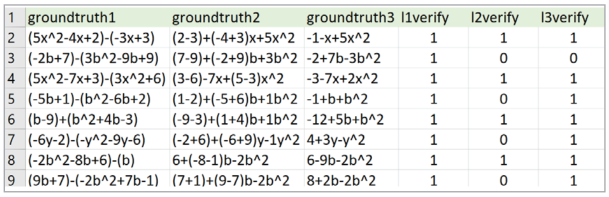
The process started with generating templates for polynomial addition, subtraction, and multiplication. Addition and subtraction templates contain 3 lines: the addition or subtraction of two polynomials; grouping the monomial coefficient of the same degree; and the simplified solution. Similarly, the multiplication template contains 4 lines. The additional line is where the multiplication derivation requires the polynomial expansion before grouping terms.

Some mistakes were intentionally added to the generated template of monomial addition and multiplication according to the types of mistakes in Table 1. First, the addition symbols were randomly replaced by subtraction, and vice versa. Next, the sums of power were randomly replaced by the product of power.

The mistakes were randomly added to the expressions at rate 5% of the total + and - signs and monomial multiplication. The ground truth, consisting of the sequence of generated MEs themselves and the correctness for each step of derivation, were saved to a CSV file as shown in Figure 9.

Figure 9

Ground truth of the addition/subtraction dataset



1	groundtruth1	groundtruth2	groundtruth3	l1verify	l2verify	l3verify
2	(5x^2-4x+2)-(-3x+3)	(2-3)+(-4+3)x+5x^2	-1-x+5x^2	1	1	1
3	(-2b+7)-(3b^2-9b+9)	(7-9)+(-2+9)b+3b^2	-2+7b-3b^2	1	0	0
4	(5x^2-7x+3)-(3x^2+6)	(3-6)-7x+(5-3)x^2	-3-7x+2x^2	1	1	1
5	(-5b+1)-(b^2-6b+2)	(1-2)+(-5+6)b+1b^2	-1+b+b^2	1	0	1
6	(b-9)+(b^2+4b-3)	(-9-3)+(1+4)b+1b^2	-12+5b+b^2	1	1	1
7	(-6y-2)-(-y^2-9y-6)	(-2+6)+(-6+9)y-1y^2	4+3y-y^2	1	0	1
8	(-2b^2-8b+6)-(b)	6+(-8-1)b-2b^2	6-9b-2b^2	1	1	1
9	(9b+7)-(-2b^2+7b-1)	(7+1)+(9-7)b-2b^2	8+2b-2b^2	1	0	1

After the ground truth was generated, symbol images were picked up and rearranged to match each of the expressions. The thickness and size of the symbols were randomized. Figure 10 shows examples of the generated images with incorrect derivations. There are 3,000 derivation images for polynomial addition and subtraction and another 3,000 images for polynomial multiplication being generated, as some examples shown in Figure 11.

Figure 10

Computer-generated images for polynomial subtraction and multiplication derivation, with incorrect steps

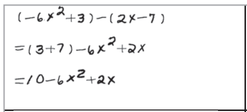
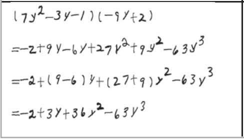
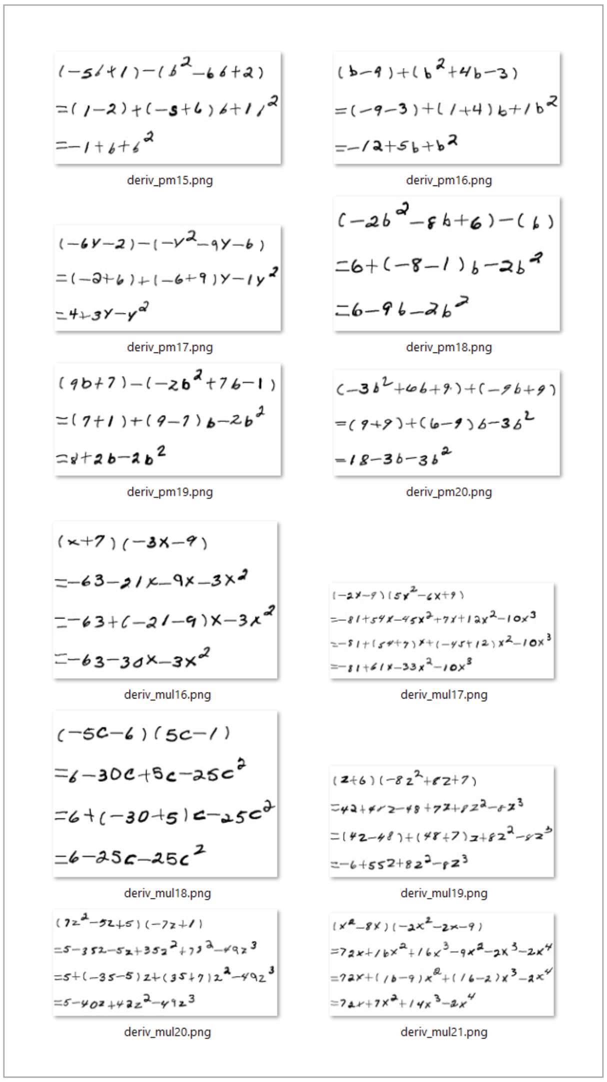
	$(-6x^2 + 3) - (2x - 7)$	
	$= (3 + 7) - 6x^2 + 2x$	Incorrect
	$= 10 - 6x^2 + 2x$	Correct
	$(7y^2 - 3y - 1)(-9y + 2)$	
	$= -2 + 9y - 6y + 27y^2 + 9y^2 - 63y^3$	Incorrect
	$= -2 + (9 - 6)y + (27 + 9)y^2 - 63y^3$	Correct
	$= -2 + 3y + 36y^2 - 63y^3$	Correct

Figure 11

Sample of generated derivation images in addition/subtraction and multiplication dataset



4.2. What Should Be Marked as "correct"?

Correctness can be defined from different perspectives: full credit for the entire derivation, or partial credit for each step of derivation. We take both of them into our evaluation. Giving full credit for the entire derivation is straightforward. Any presence of incorrect step in the derivation makes the whole thing incorrect. On the other hand, to give partial credits to student, correctness should be considered between two consecutive lines. For polynomial addition and subtraction with 3 lines, there are two transitions between lines 1 and 2, and between lines 2 and 3. Similarly for the multiplication with 4 lines, there are 3 transitions to evaluate.

4.3. Experimental Results

Tables 3 and 4 contains the number of correct partial credit marking, correct expression after the replacement when compared to the ground truth, and the CPU time for grading 3000 derivations in every setting. The maximum depth varies from 0 to 6. When maximum depth is 0, only step 1 in the expression sequence matching was utilized, i.e., the raw MER outputs were taken to the evaluation directly.

The number of correct MEs dropped slightly for the maximum depth > 5, while the number of correct marking still maintains the positive trend. It is because the more candidates allowed to use, the more expression can be replaced and become equivalent although the replacement might be incorrect.

Expression tree comparison was applied to every pair of MEs after the symbols replacement. Marking results for partial credit counts are visualized in Figure 12. There are 6,000 marks for addition and subtraction dataset, and 9,000 marks for multiplication dataset.

A short comment was generated from the comparison results. If the pair of MEs are equivalent, the comment showed "Equivalent expressions". Otherwise, the comment pointed out the part of MEs that are mismatch, as shown in Figure 13.

Table 3
Effect of depth to the correctness of expression, marking and CPU time for addition/subtraction dataset

Depth	Correct ME out of 9000	Correct marking (partial credit) out of 6000	Correct marking (full credit) Out of 3000	CPU time (sec)
0	6227	3357	813	4397
1	7872	5357	2647	8946
2	8037	5489	2693	8973
3	8035	5499	2697	9793
4	8036	5500	2696	10325
5	8040	5502	2698	11628
6	8038	5501	2698	13363

Table 4
Effect of depth to the correctness of expression, marking and CPU time for multiplication dataset

Depth	Correct ME out of 12000	Correct marking (partial credit) out of 9000	Correct marking (full credit) out of 3000	CPU time (sec)
0	7727	4689	1216	8600
1	10234	7367	2479	24930
2	10487	7571	2526	29812
3	10534	7601	2528	37719
4	10542	7606	2526	48118
5	10548	7607	2527	82183
6	10549	7613	2528	260254

Figure 12
Number of partial credit correct marking for addition and subtraction (left) and multiplication (right)

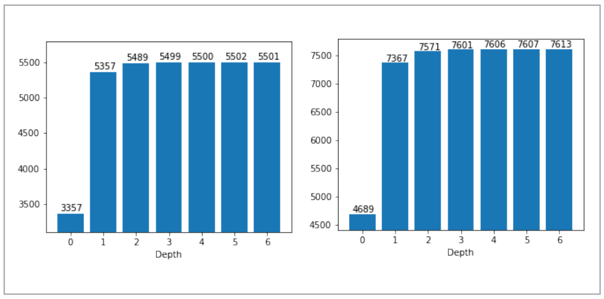
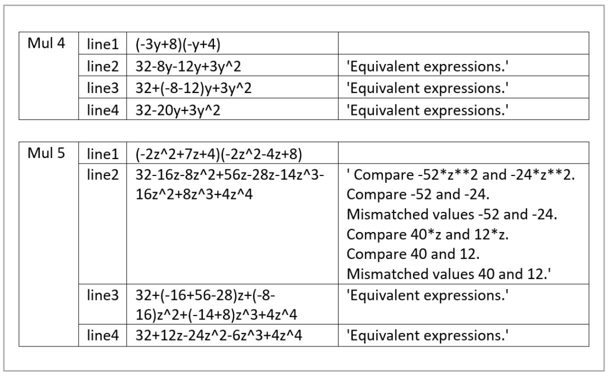


Figure 13

The system returns “Equivalent expressions” if no error is found. Otherwise, the comments for the miscalculations is reported



Mul 4	line1	(-3y+8)(-y+4)	
	line2	32-8y-12y+3y^2	'Equivalent expressions.'
	line3	32+(-8-12)y+3y^2	'Equivalent expressions.'
	line4	32-20y+3y^2	'Equivalent expressions.'

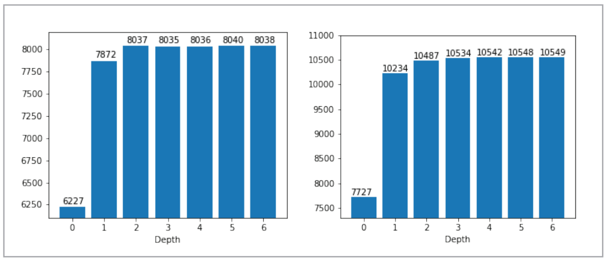
Mul 5	line1	(-2z^2+7z+4)(-2z^2-4z+8)	
	line2	32-16z-8z^2+56z-28z-14z^3-16z^2+8z^3+4z^4	' Compare -52*z**2 and -24*z**2. Compare -52 and -24. Mismatched values -52 and -24. Compare 40*z and 12*z. Compare 40 and 12. Mismatched values 40 and 12.'
	line3	32+(-16+56-28)z+(-8-16)z^2+(-14+8)z^3+4z^4	'Equivalent expressions.'
	line4	32+12z-24z^2-6z^3+4z^4	'Equivalent expressions.'

4.4. Effect of Depth to the MEs Correctness

We focused on the comparison between the $M^l(R^l)$ (the string after the replacement) and the ground truth. The correctness trend is positive as shown in Figure 14. In fact, the result when using different maximum depth greater than 1 are not significantly different.

Figure 14

Number of correct expressions for addition and subtraction dataset (left) and multiplication dataset (right)



The plot in Figure 14 shows some slight drop of correctness because the objective of the symbols replacement algorithm is to maximize the number of expression in the exact match sequence. There are exact matches caused by an incorrect replacement. For example, in Table 5, symbols replacement when using depth = 5 has the term $6a^2$, where a is replaced by 2 and the term becomes 62^2 when depth = 6. If there is a restricted format of ME, the irrelevant ME are filtered out so that the type of error can be avoided.

Table 5

Example of unexpected replacement that causes incorrect expression while maintain the equivalence of expressions between lines

Depth = 5	Depth = 6
line1 (a+2)-(6aˆ2+6a+4)	line1 (a+2)-(62ˆ2+6a+4)
line2 (2-4)+(1-6)a+6aˆ2	line2 (2-4)+(1-6)a+62ˆ2
line3 -2-5a+6aˆ2	line3 -2-5a+62ˆ2

4.5. Effect of Depth to the CPU Time

The algorithm was implemented and tested on a personal computer with CPU i7-4770, RAM 16 GB, 256 GB SATA3 SSD.

As expected, using more maximum depth showed the positive trend of correct marking while taking exponential time, as shown in Figure 15.

From the distribution of running time in Figure 16, the derivation with all correct MEs that does not need symbols replacement is the fastest group, with average time at 1.56 second per image. The second fastest group is the derivation that needs symbols replacement but return correct marking at last, with average running time at 2.17 seconds. The slowest group is the

Figure 15

Running time (in second) for 3000 images (left) addition and subtraction (right) multiplication

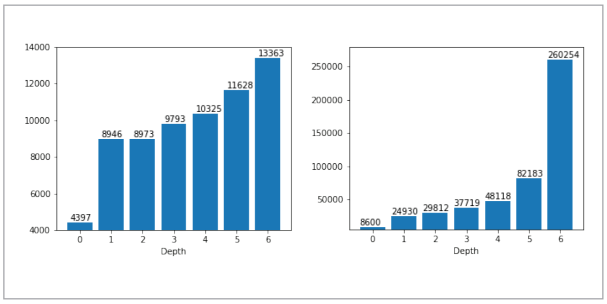
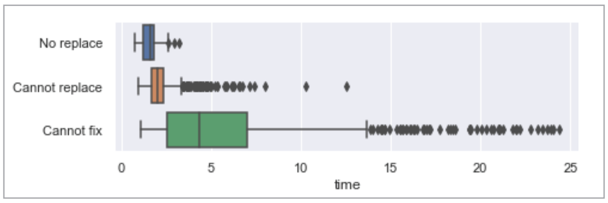


Figure 16

The distribution of running time for all 3 groups from addition/subtraction with depth = 6



derivation with mistake and cannot be fixed, with average time at 8.81 seconds. It can be seen that the last group has a wide range of variation, which depends on the number of candidate replacement.

4.6. Performance Evaluation

4.6.1. Single Symbol Accuracy

The performance of the CNN symbol recognition was tested just once because every experiment on the replacement depth shares the same symbol recognition module. There are 387,290 symbols in both datasets combined and the number of misrecognized symbols was 7,475, making single symbol recognition accuracy at 98.07%. When applied this number to the average length of MEs in the addition/subtraction and multiplication dataset at 14.60 and 21.32 symbols, respectively, the average misrecognized symbols per ME were 0.2818 and 0.4115 symbols, which are less than 1 in both datasets.

4.6.2. ExpRate

To the best of our knowledge, we cannot find any other similar system that evaluates the entire mathematical derivation after recognition, so the direct performance comparison cannot be made. However, the ExpRate (expression recognition rates) before applying symbols replacement (depth = 0) in this study (69.18% for addition/subtraction and 64.39% for multiplication dataset) were on par with the 60-65% results in most of the recent literatures that propose mathematical expression recognition systems [27, 28, 29]. ExpRate after applying the symbols replacement from our system were increased to 89.31% and 87.90% for addition/subtraction and for multiplication dataset, respectively.

4.6.3. Accuracy of Partial Credit Marking

From Tables 3 and 4, the accuracy of the partial credit marking after applying symbols replacement increases from 55.95% to 91.68% in addition/subtraction dataset, and from 52.10% to 84.58% in multiplication dataset.

4.6.4. Accuracy of Entire Derivation Marking

Instead of counting for the number correct marking in each step to give the partial credit, the other marking style is to determine the correctness of the entire derivation. From Tables 3 and 4, the accuracy of the entire derivation marking after applying symbols replacement increases from 27.10% to 89.93% in addition/subtraction dataset, and from 40.53% to 84.26% in multiplication dataset.

4.6.5. Precision, Recall, and F1-score

Precision, recall, and F1-score for the partial credit marking are shown in Tables 6 and 7. The F1-scores improve from 0.6941 and 0.6145 to 0.9494 and 0.8995 for addition/ subtraction and multiplication dataset, respectively.

4.7. Discussion

4.7.1. Optimal Replacement Depth

From Tables 6 and 7, the replacement depth 2–3 should be the optimal values of parameter since the marking results are not significantly different from the highest value, while the CPU running time is much faster than using depth = 6. This is consistent to

Table 6

Effect of depth to the precision and recall for addition and subtraction dataset

Depth	TP	TN	FP	FN	P	R	F1
0	2546	811	0	2244	1	0.5315	0.6941
1	4493	864	0	635	1	0.8761	0.9339
2	4624	865	0	506	1	0.9013	0.9481
3	4634	865	0	496	1	0.9033	0.9492
4	4636	864	1	494	0.9997	0.9037	0.9493
5	4638	864	1	492	0.9997	0.9040	0.9495
6	4638	863	2	492	0.9997	0.9040	0.9494

Table 7

Effect of depth to the precision and recall for multiplication dataset

Depth	TP	TN	FP	FN	P	R	F1
0	3249	1440	0	4076	1	0.4435	0.6145
1	5893	1474	2	1612	0.9996	0.7852	0.8795
2	6098	1473	3	1411	0.9995	0.8120	0.8961
3	6128	1473	3	1381	0.9995	0.8160	0.8985
4	6133	1473	3	1376	0.9995	0.8167	0.8989
5	6134	1473	3	1375	0.9995	0.8168	0.8990
6	6140	1473	3	1369	0.9995	0.8176	0.8995

the result of single symbol accuracy that the average misrecognized symbol is less than one, using replacement with depth = 1 already fix most of the error, and depth = 2 or 3 made a slight improvement.

4.7.2. False Negative and False Positive

For the case of false negative (the actual derivation is correct, but marked as incorrect), there are reasons that MEs cannot be fixed by the surrounding expressions to match the ground truth.

- A possible reason is the case that the longest sequence of the exact match does not contain the first line. Since the only chance to fix the first line is that it must be rebuilt with the second line reference. When the second line is incorrect, the first line has no reliable reference for correction, so the original ME cannot be restored.
- Some expressions cannot be parsed due to the existence of unexpected symbols in the way. Most of the cases is when there are broken handwritten strokes from MER, making an unwanted component in addition to the original symbols. The symbols replacement cannot unify the broken symbols, so there is no chance to restore these expressions.
- The last reason is that the lack of the candidate. If the correct replacement of the misrecognized symbol is not in the confusion matrix, the expression cannot be restored as well.

For the case of false positive, it can be seen that the symbols replacement unexpectedly replaces the incorrect derivation with a new set of symbols and becomes correct. It is the rare cases, but still is possible.

Although the false negative from the result is high, we can rely on the positive result (derivation marked as correct) because of the low false positive, which means the derivation marked as correct from the system is likely to be correct without need of further inspection. On the other hand, the derivation marked as incorrect can be either misrecognized or actually incorrect, manual inspection is strongly needed.

5. Conclusion

This research focused on designing an automatic system to help teachers more effectively grade handwritten homework assignment in the entire derivation format. The main contribution of this research is to add symbols replacement in between the MER and CAS. The candidates for replacement can be chosen from the confusion matrix and lower rank candidates of the symbol recognition.

From the results, the symbols replacement algorithm can improve F1-score of the derivation step marking from 69.41% to 94.95% for the addition/subtraction dataset and from 61.45% to 89.95% for the multiplication dataset when compared to the raw strings from MER. It can be concluded that the symbols replacement algorithm significantly improves the performance of handwritten homework grading between consecutive lines of derivation. The accuracy of marking increases as the depth of the symbols replacement increases. However, the replacement depth at 2-3 are enough to correct most of the errors.

Acknowledgement

This research project was supported by the Thailand Science Research and Innovation (TSRI grant MRG6280204), and Faculty of Science, Mahidol University, Thailand.

References

1. Afshan, N., Mehdi, S. A. An Analysis of Mathematical Expression Recognition Techniques. International Journal of Advanced Research in Computer Science, 2017, 8(5), 2021-2026. DOI: 10.26483/ijarcs.v8i5.3846
2. Chai, D. Automated Marking of Printed Multiple-Choice Answer Sheets. IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE), Bangkok, 2016, 145-149. https://doi.org/10.1109/TALE.2016.7851785
3. Chan, K., Yeung, D. Mathematical expression Recognition: A Survey. International Journal on Document Analysis and Recognition, 2000, 3, 3-15. https://doi.org/10.1007/PL00013549
4. Cipriano, B. P., Fachada, N., Alves, P. Drop Project: An Automatic Assessment Tool for Programming Assign-

ments. SoftwareX, 2022, 18. https://doi.org/10.1016/j.softx.2022.101079

5. Combéfis, S. Automated Code Assessment for Education: Review, Classification and Perspectives on Techniques and Tools. Software, 2022, 1(1), 3-30. https://doi.org/10.3390/software1010002
6. Darvishzadeh A., Entezari N., Stahovich T. Finding the Answer: Techniques for Locating Students' Answers in Handwritten Problem Solutions. 16th International Conference on Frontiers in Handwriting Recognition (ICFHR). IEEE, 2018, 587-592. https://doi.org/10.1109/ICFHR-2018.2018.00108
7. Furukori, F., Yamazaki, S., Miyagishi, T., Shirai, K., Okamoto, M. An OCR System with OCRopus for Scientific Documents Containing Mathematical Formulas, 12th International Conference on Document Analysis and Recognition, Washington, DC, 2013, 1175-1179. https://doi.org/10.1109/ICDAR.2013.238
8. Garain, U. Identification of Mathematical Expressions in Document Images. 10th International Conference on Document Analysis and Recognition, Barcelona, 2009, 1340-1344. https://doi.org/10.1109/ICDAR.2009.203
9. Hoogland, K., Tout, D. Computer-based Assessment of Mathematics into the Twenty-first Century: Pressures and Tensions. ZDM, 2018, 50. 1https://doi.org/10.1007/s11858-018-0944-2
10. Hossain, M. B., Naznin, F., Joarder Y. A., Zahidul Islam., Uddin, M. J. Recognition and Solution for Handwritten Equation Using Convolutional Neural Network. Joint 7th International Conference on Informatics, Electronics & Vision (ICIEV) and 2018 2nd International Conference on Imaging, Vision & Pattern Recognition (icIVPR), Kitakyushu, Japan, 2018, 250-255. https://doi.org/10.1109/ICIEV.2018.8640991
11. Hosseinpour, S., Milani, M., Pehlivan, H. A Step-by-Step Solution Methodology for Mathematical Expressions. Symmetry, 2018, 10, 285. https://doi.org/10.3390/sym10070285
12. Kissos, I., Dershowitz, N. OCR Error Correction Using Character Correction and Feature-Based Word Classification. 12th IAPR Workshop on Document Analysis Systems (DAS), Santorini, 2016, 198-203. https://doi.org/10.1109/DAS.2016.44
13. Le, A.D., Indurkhya, B., Nakagawa, M. Pattern Generation Strategies for Improving Recognition of Handwritten Mathematical Expressions. ArXiv, 2019, abs/1901.06763. https://doi.org/10.1016/j.patrec.2019.09.002
14. Li, X., Yue, T., Huang, X., Yang, Z., Xu, G. BAGS: An Automatic Homework Grading System Using the Pictures Taken by Smart Phones. ArXiv, abs/1906.03767, 2019.
15. Marti, U. V., Bunke, H. Using a Statistical Language Model to Improve the Performance of an HMM-based Cursive Handwriting Recognition Systems. In Hidden Markov Models. World Scientific Series in Machine Perception and Artificial Intelligence Series, Vol. 45. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2001, 65-90. https://doi.org/10.1142/9789812797605_0004
16. Marques, F. C. F. et al. Recognition of Simple Handwritten Polynomials Using Segmentation with Fractional Calculus and Convolutional Neural Networks. 8th Brazilian Conference on Intelligent Systems (BRACIS), Salvador, Brazil, 2019, 245-250. https://doi.org/10.1109/BRACIS.2019.00051
17. Nazemi, A., Tavakolian, N., Fitzpatrick, D., Fernando, C. A., Suen, C. Y. Offine Handwritten Mathematical Symbol Recognition Utilising Deep Learning. arXiv:1910.07395. [Online], 2019, Available: http://arxiv.org/ abs/1910.07395
18. Nguyen, C. T., Khuong, V. T. M., Nguyen, H. T., Nakagawa, M. CNN Based Spatial Classification Features for Clustering Offline Handwritten Mathematical Expressions. Pattern Recognition Letters, 2019, 131, 113-120. https://doi.org/10.1016/j.patrec.2019.12.015
19. Ohyama, W., Suzuki, M., Uchida, S. Detecting Mathematical Expressions in Scientific Document Images Using a U-Net Trained on a Diverse Dataset. IEEE Access, 2019, 7, 144030-144042. https://doi.org/10.1109/ACCESS.2019.2945825
20. Ramadhan, I., Purnama, B., Faraby, S. A. Convolutional Neural Networks Applied to Handwritten Mathematical Symbols Classification. 4th International Conference on Information and Communication Technology (ICoICT), Bandung, 2016, 1-4. https://doi.org/10.1109/ICoICT.2016.7571941
21. Scantron Corp. (n.d.). About Scantron. Retrieved 01 06, 2022, from Scantron website: https://www.scantron.com/company/
22. Simistira, F., Papavassiliou, V., Katsouros, V., Carayannis, G. Recognition of Spatial Relations in Mathematical Formulas. 14th International Conference on Frontiers in Handwriting Recognition, Heraklion, 2014, 164-168. https://doi.org/10.1109/ICFHR.2014.35
23. Smolinsky, L. et al. Computer-based and Paper-and-pencil Tests: A Study in Calculus for STEM Major. arXiv, 2020. https://arxiv.org/abs/2005.05462

24. Suleman, H. Automatic Marking with Sakai. Proceedings of the Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on IT Research in Developing Countries: Riding the Wave of Technology (SAICSIT'08). Association for Computing Machinery, New York, NY, USA, 2008, 229-236.https://doi.org/10.1145/1456659.1456686

25. Sukkarieh, J., Pulman, S., Raikes, N. Auto-marking: Using Computational Linguistics to Score Short, Free Text Responses. International Association for Educational Assessment (IAEA), 2003.

26. Wang, X., Gülenman T., Pinkwart, N., de Witt, C., Gloerfeld, C., Wrede, S., Automatic Assessment of Student Homework and Personalized Recommendation. IEEE 20th International Conference on Advanced Learning Technologies (ICALT), 2020, 150-154. https://doi.org/10.1109/ICALT49669.2020.00051

27. Wu, C., Du, J., Li, Y., Zhang, J., Yang, C., Ren, B., Hu, Y. TDv2: A Novel Tree-Structured Decoder for Offline Mathematical Expression Recognition. Proceedings of the AAAI Conference on Artificial Intelligence, 2022, 36(3), 2694-2702. https://doi.org/10.1609/aaai.v36i3.20172

28. Yuan, Y., Liu, X., Dikubab, W. Liu, H., Ji, Z., Wu, Z., Bai, X. Syntax-Aware Network for Handwritten Mathematical Expression Recognition. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022, 4553-4562. https://doi.org/10.1109/CVPR52688.2022.00451

29. Zhao, W., Gao, L., Yan, Z., Peng, S., Du, L., Zhang, Z. (2021). Handwritten Mathematical Expression Recognition with Bidirectionally Trained Transformer. Document Analysis and Recognition - ICDAR 2021, 2021. https://doi.org/10.1007/978-3-030-86331-9_37