Information Processing and Management 60 (2023) 103088

Contents lists available at ScienceDirect

Information Processing and Management

journal homepage: www.elsevier.com/locate/ipm

A methodology for character recognition and revision of the linear equations solving procedure

María Cristina Guevara Neri [a], Osslan Osiris Vergara Villegas [a,*], Vianey Guadalupe Cruz Sánchez [a], Humberto de Jesús Ochoa Domínguez [a], Manuel Nandayapa [a], Juan Humberto Sossa Azuela [b]

[a] *Universidad Autónoma de Ciudad Juárez, Instituto de Ingeniería y Tecnología, Avenida del Charro 450 Norte, Partido Romero, P.C. 32310, Juárez, Chihuahua, Mexico*

[b] *Instituto Politécnico Nacional, Centro de Investigación en Computación, Av. Juan de Dios Bátiz S/N, Nueva Industrial Vallejo, P.C. 07738 CDMX, Mexico*

ARTICLE INFO

Keywords:
Linear equations
Convolutional neural network
Character recognition
Equation solving procedure

ABSTRACT

Linear equations are valuable for real-world modeling phenomena involving at least one variable. However, verifying if the procedure followed by a human for solving a linear equation was done correctly is still a complicated matter. In this paper, we propose a methodology for the automatic character recognition and revision of the solving procedure of linear equations with one unknown. First, a camera is used to acquire an image of the handwritten solving procedure. Then, the image is pre-processed, and each character and equation lines are segmented. Subsequently, a convolutional neural network (CNN) is used to conduct the character recognition stage. Finally, a comparison rule is applied to revise the solving procedure. The character recognition was verified on a 2800 image data set (2100 for training and 700 for testing), including the ten digits and four symbols: ×, +, -, /. The revision procedure was tested on a data set with 140 handwritten equations (125 for training and 15 for testing). The results revealed that we recognized handwritten characters with an accuracy of 99%, which is similar to the state-of-the-art. Moreover, our proposal revised the solving procedure with an efficiency of 86.66%.

1. Introduction

Nowadays, information and communication technologies (ICTs) play an essential role in learning. Any learning mediated by a computer to provide educational instruction is known as computer-assisted learning (CAL) (Alabdulakareem & Jamjoom, 2020). Frequently, students use CAL to acquire knowledge regarding a foreign language (Mei et al., 2018), medicine (Guimaraes et al., 2019), or mathematics (Tokac et al., 2019).

Cross-cultural research results related to student academic performance have shown that mathematics's failure rate is higher than in other areas. Additionally, learning mathematics produces unsatisfactory academic results and causes stress and anxiety in students (Coronado, 2017). Therefore, there is a significant interest in studying the impact of technological tools in teaching and learning mathematics (Cullen et al., 2020; Thurm & Barzel, 2022; Verbruggen et al., 2021).

* Corresponding author.
E-mail addresses: al171517@alumnos.uacj.mx (M.C. Guevara Neri), overgara@uacj.mx (O.O. Vergara Villegas), vianey.cruz@uacj.mx (V.G. Cruz Sánchez), hochoa@uacj.mx (H.d.J. Ochoa Domínguez), mnandaya@uacj.mx (M. Nandayapa), humbertosossa@gmail.com (J.H. Sossa Azuela).

https://doi.org/10.1016/j.ipm.2022.103088
Received 14 May 2022; Received in revised form 21 August 2022; Accepted 1 September 2022
Available online 19 October 2022

Cullen et al. (2020) suggested that digital tools, used appropriately, can contribute positively to solving problems associated with learning mathematics. Therefore, the field of mathematics has benefited from using CAL tools (Meeter, 2021; Wook et al., 2020).

The advance achieved by current technologies has made possible the creation of programs capable of solving mathematical expressions with reduced processing workload by using graphic processing units (GPUs) (Lin et al., 2017). For example, an intelligent device could resolve algebraic problems by entering data through touch screens or scans where the image is processed by a system capable of recognizing handwritten characters. Likewise, it is common to find applications for the direct solution of mathematical exercises corresponding to different areas (algebra, calculus, and statistics), which try to assume the role of a teacher.

Today, it is common to have applications that use an image of the handwritten problem to solve mathematical problems involving different operations types (Zhang, Wang et al., 2020). These applications have a structure composed of the following stages: (i) pre-processing, (ii) segmentation, (iii) extraction, (iv) recognition, and (v) solving of expressions (Shuvo et al., 2021).

Currently, there are systems for recognizing and solving handwritten mathematical expressions (presenting the correct solution as an answer). However, it could be helpful to build an application that allows the user to propose the solving of a problem and only request support from the application to review the solving proposed procedure. Furthermore, revising the solving process is necessary to detect any details that lead to an erroneous result and ensure student understanding. To the best of our knowledge, there is no sign of similar work in the literature. Therefore, the main contributions of this paper are as follows:

1. We propose a method to segment individual equations in a solving procedure composed of more than one line.
2. We propose a method based on comparison rules to revise a set of consecutive linear equations corresponding to the same solving procedure.
3. We propose a methodology for the automatic review of the procedure for solving linear equations with one unknown.

The rest of the paper is organized as follows. Section 2 presents the research objectives. Section 3 presents the literature review and the theoretical background for solving linear equations. Section 4 shows a description of the proposed methodology. The experiments and results are presented in Section 5. Finally, the conclusions and future work are shown in Section 6.

2. Research objectives

Although the literature is vast regarding manuscripts to recognize (i) digits (Ahlawat et al., 2020; Ali et al., 2019; Alvear et al., 2019; Baldominos et al., 2019), (ii) mathematical symbols (Kukreja, 2021; Kukreja & Sakshi, 2022), and (iii) handwritten mathematical expressions (Aggarwal et al., 2022; Duc, Indurkhya et al., 2019; He et al., 2020; Huang et al., 2020; Wu et al., 2020; Zhang, Moucheré et al., 2020; Zhelezniakov et al., 2021), there is no sign of work in recognizing and revising mathematical expressions. Therefore, this paper aims twofold. First, to recognize handwritten mathematical first-degree algebraic equations with one unknown. Second, to revise the procedure for solving first-degree algebraic equations with one unknown.

3. Literature review and theoretical background

This section presents the literature review regarding recognizing handwritten digits and mathematical expressions and the theoretical background for solving linear equations.

3.1. Literature review

Mathematical symbol recognition is crucial in recognizing handwritten mathematical expressions. Moreover, the review of the solving procedure could be conducted only after mathematical expression recognition.

This work addressed the one unknown linear equations solving procedure. As explained in Section 4.4.2, the ten digits and the four symbols: $\times, +. -, /$ should be recognized. Therefore, we searched for the newest papers regarding handwritten digit recognition.

On the other hand, we searched for papers devoted to mathematical expression recognition and revision of linear equation solving procedures. As a result, there are no signs of papers devoted to these tasks. However, we analyzed the newest works regarding the mathematical expression recognition problem.

Many research studies have been published about recognizing handwritten digits. Interested readers can consult the work by Baldominos et al. (2019) to obtain a comprehensive overview of this field. The review revealed that handwritten digit recognition works could be divided into traditional and deep learning approaches. The traditional approaches use support vector machines (SVM), random forest (RF), or artificial neural networks (ANN). On the other hand, most deep learning approaches employ a convolutional neural network(CNN) as the backbone and propose architecture additions.

We collected only journal papers from 2018-to mid-2022. Therefore, the papers from conferences were discarded. It is important to highlight that no papers employing traditional approaches were detected in this period. Moreover, intending to present comparisons to our proposal, we only analyzed the papers that employed the modified national institute of standards and technology (MNIST) data set (LeCun et al., 2018).

MNIST includes 70,000 handwritten digits. The 28×28 grayscale images were centered by computing the center of mass of the digit and translating it to the image center position.

A summary of the analyzed papers' features is shown in Table 1. As can be observed, most of the papers employed a CNN as a recognition method. The works of Albahli et al. (2020) and Alghazo et al. (2019) employed different recognition methods and

Table 1
Summary of 17 experimental studies for digit recognition.

Authors	Recognition method	Training/Testing samples	Accuracy (%)
Qiao et al. (2018)	Q-ADBN	60 000/10 000	99.18
Kulkarni and Rajendran (2018)	SNN	50 000/10 000	98.17
Dash et al. (2018)	PSD	60 000/10 000	99.11
Alvear et al. (2019)	SDAE	50 000/10 000	100
Ali et al. (2019)	CNN	60 000/5130	99.21
Alghazo et al. (2019)	AIRS, MLP, LR and RF	60 000/10 000	97.30
Min et al. (2020)	PEO-FOBP	60 000/10 000	96.54
Ali et al. (2020)	CNN-ELM	4478 (only tests)	99.8
Jha and Cecotti (2020)	GAN	60 000/10 000	99.53
Aly and Almotairi (2020)	DCSOM	60 000/10 000	99.43
Albahli et al. (2020)	CNN, RNN, MLP, SNN and KT	60 000/10 000	100.00, 53.68, 98, 99.61 and 99.31
Ahlawat et al. (2020)	CNN	60 000/10 000	99.89
Zhao and Liu (2020)	KNN+RF	60 000/10 000	98.1
Madakannu and Selvaraj (2020)	CNN	60 000/10 000	99.11
Abdullah et al. (2021)	CNN	42 000/28 000	99.98
Urazoe et al. (2021)	CNN	60 000/10 000	99.61
Saqib et al. (2022)	CNN	60 000/10 000	99.64
Our proposal (2022)	**CNN**	**42 000/28 000**	**99.45**

*Q-ADBN: Adaptive Q-learning deep belief network, SNN: Spiking neural network, PSD: Perceptual shape decomposition, SDAE: Stacked denoising auto-encoding, CNN: Convolutional neural network, AIRS: Artificial immune recognition system, MLP: Multilayer Perceptron, LR: Logistic regression, RF: Random forest, PEO-FOBP: Population extremal optimization-fractional order Backpropagation, CNN-ELM: Convolutional neural network-extreme learning machine, GAN: Generative adversarial networks, DCSOM: Deep convolutional self-organizing maps, RNN: Recurrent neural network, SNN: Siamese neural network, KT: Knowledge transfer, KNN: K-nearest neighbor.

selected the one with the best performance. The MNIST data set is split for training and testing in different quantities, mostly 60,000 and 10,000 samples, respectively. Regarding the accuracy, the works by Albahli et al. (2020) and Alvear et al. (2019) reported a 100%. However, a hundred percent in recognition model is controversial for the pattern recognition community. Therefore, the MNIST data set classification problem is still an open challenge. The mean accuracy of the 17 works is 99.09%.

Regarding handwritten mathematical expression recognition (HMER), we searched for the newest journal and conference papers from 2018-to mid-2022. We recommend the paper of Zhelezniakov et al. (2021) to obtain a wide panorama regarding HMER. Mathematical expression recognition could be conducted in two ways, offline and online. Offline recognition takes a static image of the expression as input, while online recognition uses a dynamic representation such as pen traces or finger movements.

The workflow for HMER comprises the stages of (i) preprocessing, (ii) segmentation, (iii) classification and (iv) structural analysis. However, the newest techniques based on deep learning methods implicitly perform the segmentation stage.

A summary of the HMER papers analyzed is shown in Table 2. As can be observed, in the last four years, most HMER methods have been based on deep neural networks. No work was detected that employed traditional techniques. The mean recognition accuracy of the twelve works analyzed is only 50.17%.

The competition on recognition of online handwritten mathematical expressions (CROHME) was the main data set employed for testing HMER algorithms. CROHME provides more than 10,000 expressions handwritten by hundreds of writers from different countries (Kukreja, 2021).

Many methods were based on the encoder–decoder framework (Zhang et al., 2019, 2021). Frequently, a CNN was employed as an encoder, and an attention mechanism based on a recurrent neural network (RNN) was used as the decoder. The encoder reads a source and encodes it into a fixed-length vector. The decoder outputs the translation of the encoded vector.

The main advantage of deep learning methods is that feature extraction is not hand-made performed. Moreover, most deep learning methods do not need the segmentation stage. The main disadvantage of most deep learning-based methods is the need for hundreds of images, powerful processors, and expensive GPUs. Moreover, the training is time-consuming, which implies that it takes several hours or even days.

We cannot conduct comparisons of our work with CROHME because no linear equation expressions were included. Moreover, as can be observed from Table 2 even when powerful deep learning-based methods were employed, the recognition rate was still low. Consequently, we have preferred traditional computer vision methods as a base of our proposal, and deep learning was used only for the digit recognition stage.

3.2. Theoretical background

According to Baldor (2019), an equation is an equality in which there are one or more unknown quantities, called variables, where the equality is only true for specific values of the unknowns. If the equation contains a single variable whose exponent equals one, it is known as a first-degree equation or linear equation in one variable. The solving procedure of this type of equation will be the main focus of this paper.

Solving an equation means finding the root or solution for which the equation becomes true. A linear equation in one variable has only one solution. The following axiom must be considered to solve an equation.

Table 2
Summary of 12 experimental studies for HMER.

Authors	Recognition method	Data set	Accuracy (%)
Minh et al. (2018)	CYK	CROHME 13	32.19
		CROHME 14	32.86
		CROHME 16	43.94
Zhang et al. (2019)	EENN	CROHME 14	61.16
		CROHME 16	57.02
Duc, Indurkhya et al. (2019)	SON	CROHME 14	37.63
Duc, Dai et al. (2019)	PGS	CROHME 14	48.78
		CROHME 16	45.60
Chan (2020)	Oversegmentation	CROHME 14	58.22
		CROHME 16	65.65
		CROHME 19	65.22
Wu et al. (2020)	PAL	CROHME 14	54.87
Zhang, Moucheré et al. (2020)	BLSTM	CROHME 14	29.91
		CROHME 16	27.03
Wang et al. (2021)	SCAN	CROHME 13	58.11
		CROHME 14	54.29
		Chinese data	97.16
Zhang et al. (2021)	TSBD	CROHME 14	50.4
		CROHME 19	50.6
Pal and Singh (2022)	R-GRU	CROHME 14	43.72
		CROHME 16	41.19
Sakshi and Kukreja (2022a)	SVC	Kaggle handwritten math	89.76
Sakshi and Kukreja (2022b)	OpenCV segmentation and contour detection	Aida	94.30

*CYK: Cocke–Younger–Kasami algorithm, EENN: End-to-end neural network, SON: Stroke order normalization, PGS: Pattern generation strategies, PAL: Paired adversarial learning, BLSTM: tree-based long short-term memory network, SCAN: Stroke constrained attention network, TSBD: Tree structure based decoder, R-GRU: Regularization-gated recurrent unit, SVC: Support vector classifier.

Fundamental Axiom of Equations. *If equal operations are verified with equal amounts, the results will be equal.*

Two rules are derived from the fundamental axiom of equations:

1. If the same quantity, positive or negative, is added to the two members of an equation, the equality subsists.
2. If the two members of an equation are multiplied by the same quantity, positive or negative, the equality remains.

Solving an equation consists of transposing terms on both members until the variable's value is found. In summary, both rules are applied repeatedly to a linear equation to find its solution. Fig. 1 shows an example of the procedure to solve a linear equation in one variable.

The first step verifies if any operation can be performed before starting the transposition of terms. If there are no like terms to combine, as shown in Fig. 1(a), then it is decided which side of the equation the variable will be left. The variable is placed on the left side. Hence, the equation reading coincides with the ordinary reading language "x equals 8". The variable goes first (left), and the numerical value comes next (right).

In the second step, a transposition of terms is applied. The transposition is performed by clearing the variable from the left side and adding the additive inverse (a number plus its additive inverse is equal to 0) of the constant term from the left side on both equation members. Rule 1 and the fundamental axiom are used, as shown in Fig. 1(b).

A reduction of like terms is applied in step 3, as shown in Fig. 1(c). Then, a transposition of terms is conducted in step 4, as illustrated in Fig. 1(d). The process eliminates the literal term from the right side and adds the additive inverse of the literal term from the right side on both equation members. In step 5, a reduction of like terms is applied, as shown in Fig. 1(e).

Afterward, the variable is cleared by multiplying both members of the equation by the multiplicative inverse (a number multiplied by its multiplicative inverse is equal to 1) of the coefficient of the literal term, as shown in Fig. 1(f). Finally, a simplification is performed to obtain the solution of the equation, as seen in Fig. 1(g).

In the end, the solution can be verified by substituting the value obtained in the original equation. The solution is correct if the equality is true, as illustrated in Fig. 1(h).

4. Character recognition and revision of the linear equation solving procedure

Our proposal was implemented on a mid-range computer (Intel Core i5 with 8 Gb RAM). Nevertheless, shortly, we want to migrate our proposal for running on mobile devices.

$$3x - 4 = x + 6$$

(a) Initial linear equation in one variable

$$3x - 4 + 4 = x + 6 + 4$$

(b) First application of rule number 1

$$3x = x + 10$$

(c) Reduction of like terms

$$3x - x = x + 10 - x$$

(d) Second application of rule number 1

$$2x = 10$$

(e) Reduction of like terms

$$\left(\frac{1}{2}\right)2x = (10)\left(\frac{1}{2}\right)$$

(f) Application of rule number 2

$$x = 5$$

(g) Root or solution of the equation

$$3(5) - 4 = (5) + 6$$
$$15 - 4 = 5 + 6$$
$$11 = 11 \checkmark$$

(h) Solution verification

Fig. 1. Example of the solving procedure of a linear equation in one variable.

As can be observed from Table 1 the accuracy for digit recognition using deep learning methods is high (close to 100%). Therefore, we employed the LeNet architecture for the character recognition stage. The LeNet can be executed on a computer with low to mid-range features.

On the other hand, as can be observed from Table 2 most deep learning methods do not need to execute all of the stages for a traditional HMER. However, the accuracy is still low (less than 60%). Since using deep learning does not increase accuracy, we prefer to implement the traditional stages of a computer vision system that do not demand many computational resources. Traditional computer vision stages are suitable to be implemented in mobile devices.

The proposed methodology is shown in Fig. 2, and each stage is described in the following subsections.

4.1. Data set creation

Many data sets of mathematical expressions have been presented in the literature (Anitei, 2020). However, we considered a different alternative using real-life examples of equations.

We ask 125 undergraduate students from a Mexican university to solve a linear equation with one variable. We established seven limitations, considering that students can write and solve the equations differently:

• The equations must be written legibly on a sheet of white paper and using a pencil or pen with black or blue ink.

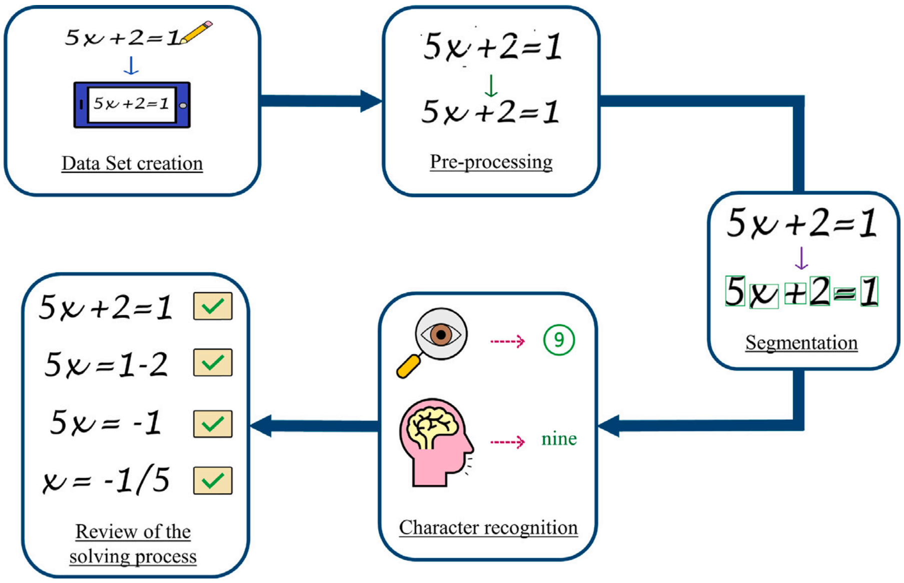


Fig. 2. Methodology for reviewing the solving procedure of a linear equation with one variable.

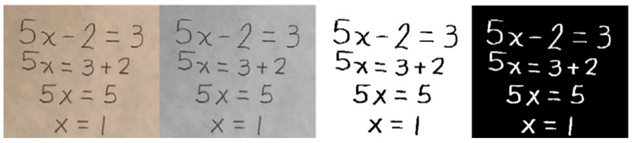


Fig. 3. Color plane conversion. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

- The equations must not contain crossed-out or overlapping elements.
- The letter x must represent the unknown.
- The symbol used to represent a division must be the forward-slash "/".
- The multiplying operations must be implicit (there should be no multiplication symbols).
- Quantities must be represented rationally by definition as an integer or a fractional number (decimal numbers are not allowed).
- During the solving procedure, each new equation must be written as a new line consecutively down.

The images for the solving procedure were acquired with a 12 MP cell phone camera, without flash, and in a room with a traditional light bulb. The resolution of the device used to acquire the images is important for pre-processing the acquired images.

4.2. Pre-processing

Each of the 125 images is pre-processed in three stages: (i) color plane conversion, (ii) morphological dilation, and (iii) denoising.

4.2.1. Color plane conversion

The acquired RGB color image is converted to grayscale by a weighted sum of the Red, Green, and Blue components, as shown in Eq. (1).

$$0.2989 * R + 0.5870 * G + 0.1140 * B \tag{1}$$

Then, Otsu's method (Yu et al., 2021) is used to calculate a threshold to binarize the grayscale image. Otsu's method chooses a threshold that minimizes the intraclass variance of the thresholded black and white pixels. Hence, pixel values greater than the threshold are set to one and the remaining zero.

Finally, the colors are inverted (black for the background and white for the objects). The results of the color plane conversion are shown in Fig. 3.

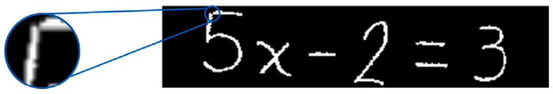


Fig. 4. Character with incomplete strokes.

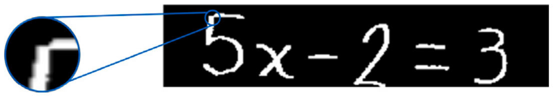


Fig. 5. Character with continuous strokes.

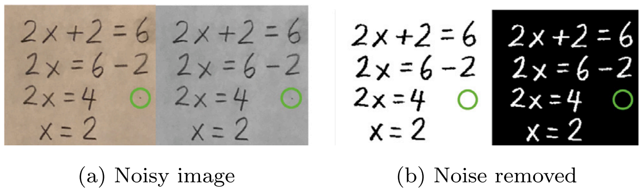


Fig. 6. Results for image denoising.

4.2.2. Morphological dilation

Morphological dilation expands objects in an image by adding pixels to the inner and outer boundaries. Dilation involves the binary image and a structuring element. The structuring element is a matrix containing zeros and ones. The size of the structuring element is defined by the matrix dimension, and the pattern of zeros and ones determines the shape. The origin of the structuring element is placed on each image pixel. If any part of the structuring element overlaps with an image object, the pixel is added to create the dilated image. The dilation process is described by Eq. (2).

$$\mathbf{A} \oplus \mathbf{B} = \{z|(\mathbf{B})_z \cap \mathbf{A} \neq \emptyset\} \tag{2}$$

where **A** is the binary image and **B** is the structuring element. The dilation of **A** and **B** is defined as the set of pixel locations z, where the structuring element overlaps with foreground pixels in the binary image when translated to z.

After binarization, some characters in the equation might have incomplete strokes, as shown in Fig. 4. Therefore, we applied dilation to connect strokes only a few pixels apart.

Dilation was performed using a linear structuring element in the form of a horizontal line. After running several experiments, we found a relationship between image resolution and the size of the structuring element.

If we use a small fixed value for the size of the structuring element, the strokes in the high-resolution images were not "filled enough". On the other hand, if we use a big fixed value, the characters in the low-resolution images are completely lost. Consequently, we set two different fixed values: 4 for the low-resolution images and 30 for the high-resolution images.

When dilation is applied to the image, generates the *thickening* of the characters, as shown in Fig. 5. The thickening can be observed when Figs. 4 and 5 are compared.

4.2.3. Denoising

In the dilated image, unwanted objects (noise) can appear and cause errors in the following methodology stages. Therefore, denoising must be carried out. We established that all 8-connected components (objects) from the image smaller than an x number of pixels are removed based on the image resolution.

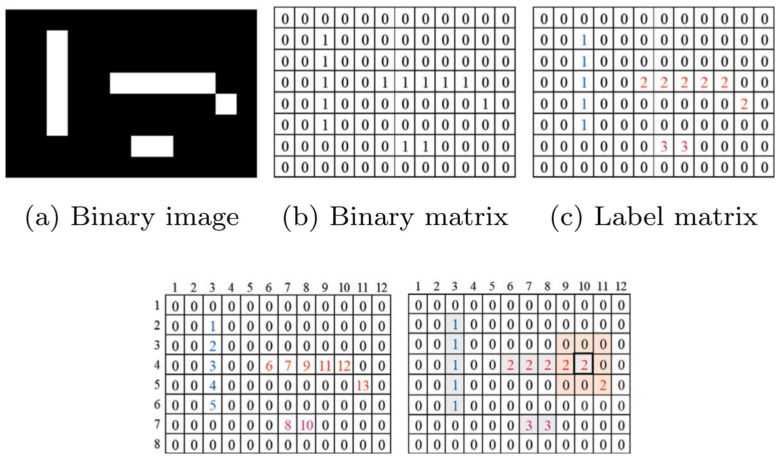
The procedure to determine the x value of the maximum number of pixels to be eliminated is shown in Algorithm 1. Moreover, Fig. 6 shows an example of the denoising procedure by applying Algorithm 1 to an image.

Algorithm 1 Calculus of the maximum number of pixels to eliminate

```
1: if the image resolution is smaller than 8 MP then
2:     x is equal to 0.02 percent of the image
3: else
4:     x is equal to 0.01 percent of the image
5: end if
```

4.3. Segmentation

Segmentation is the task of splitting an image into non-overlapped areas. As a result, a label is assigned to each image pixel. In the literature, accurate segmentation deep learning-based techniques were reported (Aghalari et al., 2021; Arias, 2022; Lu et al.,



(a) Binary image (b) Binary matrix (c) Label matrix

(d) Pixel labeling (e) 8-connected objects

Fig. 7. Example of the labeling process performed in a binary image.

2021). Unfortunately, most of these techniques need a massive quantity of images, the process of training is time-consuming and needs powerful hardware resources. Furthermore, the stage of labeling each object to be learned is also demanding in terms of time and effort.

Considering the disadvantages of deep learning methods, we decided to implement traditional computer vision segmentation. Segmentation is performed in two stages: (i) segmentation of individual characters and (ii) segmentation of individual equations.

4.3.1. *Segmentation of individual characters*

First, all objects in the image are labeled. Then, the strongly connected objects are determined. Hence, each object is extracted as an individual image.

A label matrix containing labels for the components was created. Each label was assigned to each object in the binary image. The pixels labeled 0 are the background, the pixels labeled 1 make up the first object, the pixels labeled 2 make up the second object, and so on. Fig. 7 shows an example of an 8 × 12 binary image (Fig. 7(a)), its binary matrix (Fig. 7(b)), and its label matrix (Fig. 7(c)).

The labeling of the objects consists of reviewing each pixel. From top to bottom and from left to right, a label is assigned to the 8-connected objects. Fig. 7(d) shows an example of the order in which the thirteen pixels with value one are reviewed. 8-connected means that the pixels are connected if their edges or corners touch. Therefore, two adjacent pixels are part of the same object if they are both on and are connected along the horizontal, vertical, or diagonal direction. Fig. 7(e) illustrates why the thirteenth pixel (row 5, column 11) is labeled as “2” as it is part of the 8-connected neighborhood formed by the twelfth pixel previously labeled as “2”.

After labeling, the strongly connected components of the sparse matrix corresponding to the label matrix were determined. Fig. 8 shows an example of the process of finding the strongly connected components of an 8 × 12 binary image (Fig. 8(a)), its binary matrix (Fig. 8(b)), its label matrix (Fig. 8(c)), its sparse matrix (Fig. 8(d)), and equivalence graph (Fig. 8(e)). Performing this process is important because it is possible to find two or more labeled components in the same object, as shown in Fig. 8, where there are only two different objects in the image, but the assigned labels are four.

The Dulmage–Mendelsohn decomposition is performed to determine the connected components of the sparse matrix. The procedure transforms the sparse matrix into upper block triangular form through row and column permutations, each block corresponding to a connected component. In the example shown in Fig. 8(d), two blocks can be formed in the sparse matrix, which translates into two main labels (Fig. 8(f)) that coincide accordingly with what we can visually appreciate in the image.

Finally, each object is extracted as an individual image. Once all the pixels of an object are known, the area identification is conducted by delimiting the object in a rectangular way through the vertices (the pixels of the object that are farthest to the left, top, bottom, and right), as shown in Fig. 9. Each extracted object will be needed for the segmentation of individual equations.

4.3.2. *Segmentation of individual equations*

The method consists of splitting the equations according to the position of the previously segmented objects in the image. First, the edges of the whole binary image are cropped and individual objects are detected, as shown in Fig. 10.

Then, the image is analyzed under the following proposal:

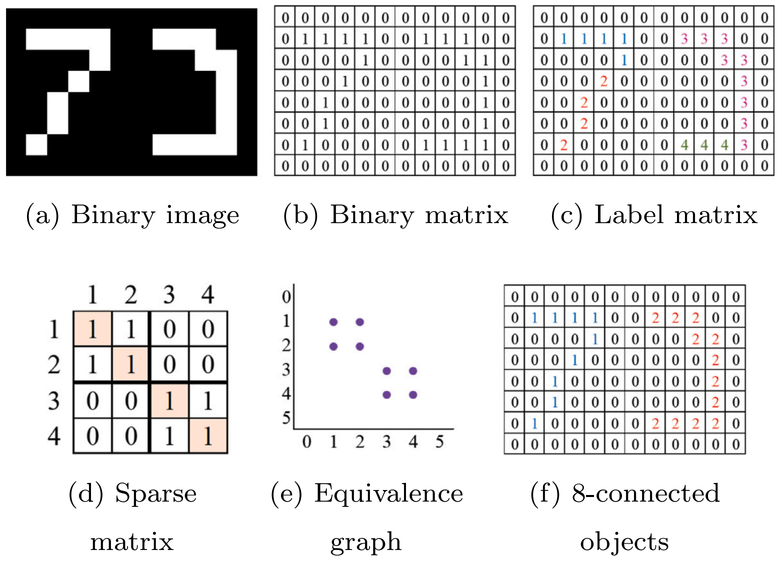


Fig. 8. Example of strongly connected objects determined in a binary image.

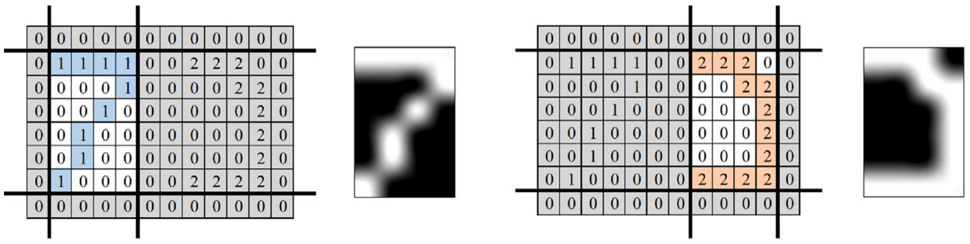


Fig. 9. Example of the segmentation of individual objects in a binary image.

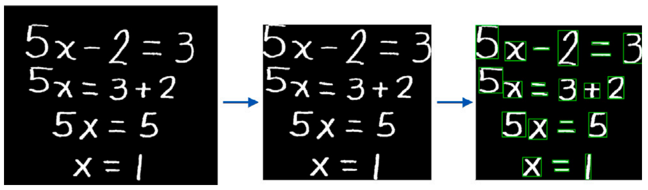


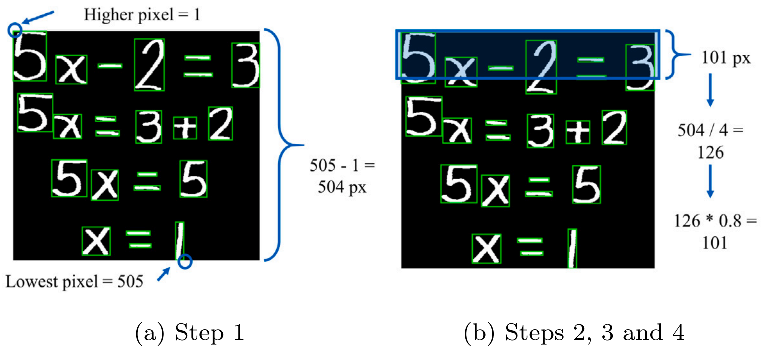
Fig. 10. Cropping of the edges and detection of objects in the binary image.

1. Measure the overall height of the cropped image.
2. Count the number of individual equations within the image.
3. Divide the total height by the number of equations to estimate the height of each equation.
4. Create a *window* of 80% of the height of each equation and determine if the highest pixel of a previously segmented object corresponds to the first equation.
5. Crop the image again, leaving out the first equation.
6. Repeat the process performed with the first equation with the rest of the equations.

Fig. 11 shows an example of the previous procedure applied to the second image of Fig. 10 with the first two equations (lines).

4.4. Character recognition

A wide variety of methods has been proposed for character recognition, such as deep learning (Boufenar et al., 2018; Ptucha et al., 2019; Weng & Xia, 2020; Zhang et al., 2017), support vector machines (Phangtriastu et al., 2017; Shinde et al., 2021), and adversarial learning (Wu et al., 2020). We performed the character recognition stage with a convolutional neural network (CNN).



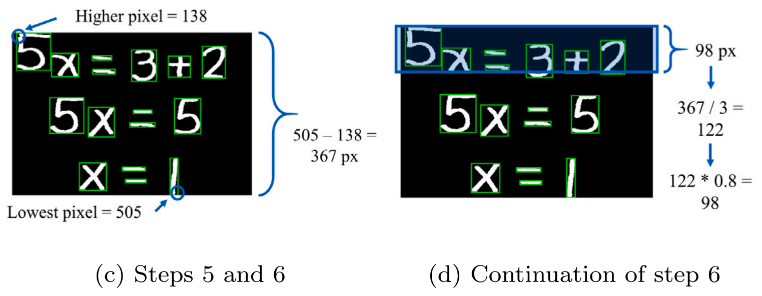


Fig. 11. Example of the segmentation of individual equations.

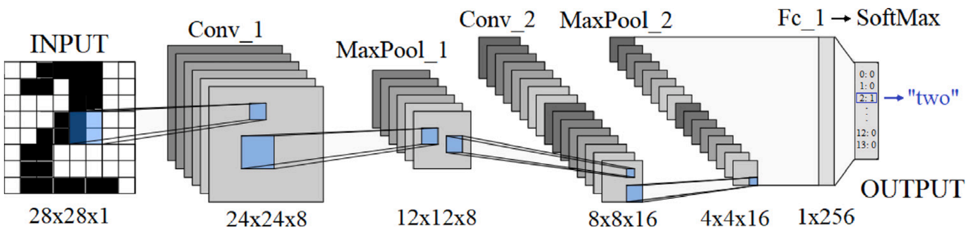


Fig. 12. CNN implemented for character recognition.

4.4.1. CNN architecture

In our past work (Guevara, 2021), we conducted an ablation study to determine the architecture of the CNN employed. The results showed that the architecture of the LeNet presented in Fig. 12 offers better results (LeCun et al., 1998). Each layer is described below.

INPUT. It represents the input image to be classified. The size of the input is 28 × 28 pixels.

Conv_1. It is the first convolutional layer of the network. Eight filters of 5 × 5 pixels are applied to the input image with a stride of 1 and a padding of 0. Then, a ReLU activation layer is applied to produce a volume of feature maps of the size of 24 × 24 × 8, where 24 × 24 is the size of each map after convolution and 8 is the number of feature maps yielded by the filters.

MaxPool_1. In this layer, a pooling operation is performed that calculates the maximum or largest value in each patch of each feature map. A window size of 2 × 2 is defined for the rectangular reduction region with a stride of 2. The output volume is of the size of 12 × 12 × 8, where 12 × 12 is the size of the reduced data, and 8 is the number of feature maps.

Conv_2. It is the second convolutional layer of the network. Sixteen filters of 5 × 5 pixels are applied to the output of the previous layer with a stride of 1 and a padding of 0. Then, a ReLU activation layer is applied to produce a volume of feature maps of size 8 × 8 × 16.

MaxPool_2. A window size of 2 × 2 is defined for the rectangular reduction region with a stride of 2. The output is a size of 4 × 4 × 16. Before the fully connected layer, the resulting convolutional layer outputs are flatted to produce 256 features.

Fc_1. It is a fully connected layer that has 256 inputs which are the result of the mathematical product of the output given by the previous flattening operation (4 × 4 × 16 = 256). The output size is 14, which is equal to the desired number of classes.

SoftMax. The softmax layer is implemented right before the output layer, it assigns decimal probabilities to each class in a multi-class problem. The output of this layer is a vector of size 14 × 1, which is formed by positive values (whose sum is equal to 1).

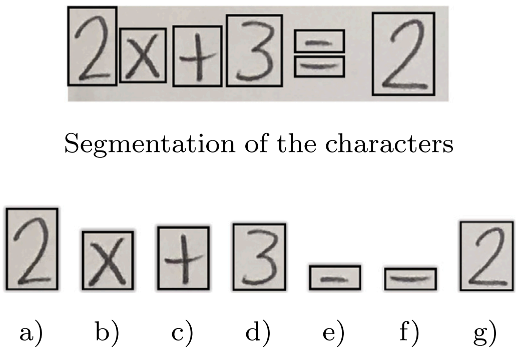


Fig. 13. Segmentation and type of extracted characters.

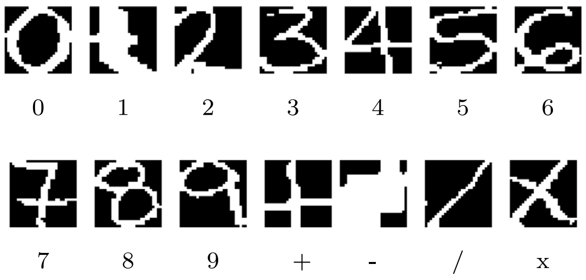


Fig. 14. Example of a character from each class.

ClassOutput. After the set of probabilities is obtained, the class with the highest probability is activated and the image is classified.

A grid search was conducted for the stage of hyperparameter tuning. The resulting hyperparameters are shown below.

Optimizer: Stochastic gradient descent with momentum (SGDM).

Momentum: 0.9 (standard value).

Initial learning rate: 0.01.

Maximum number of epochs: 100.

Mini-batch size: 128 (standard value).

Validation frequency: 20.

We tested our network with the standard MNIST database (LeCun et al., 2018), to verify the performance. As a result, we obtained an accuracy of 99.45%, which is similar to the results shown in Table 1

4.4.2. CNN training and validation

The 125 images of the data set were pre-processed following the explanation presented in Section 4.2. Then, each character was manually segmented using Microsoft Paint.

It is important to highlight that each extracted character has a different resolution. Therefore, its size is standardized to 28 × 28 pixels. Fig. 13 shows the character segmentation performed on an equation and the three types of characters extracted: (a), (d) and (g) are digits, (b) is a variable, and (c), (e) and (f) are symbols. It is appropriate to clarify that the comparison operator "=" will not be considered for the moment. That is, there will be no data set corresponding to that character. This will be clarified later.

In the end, we obtained 2800 images of 14 different character types drawn from a set of 125 different equations. In other words, we got 200 images of each character per class.

Fig. 14 shows an example from every character class. The classes are the ten digits, the addition, subtraction, and division symbols, and the symbol x, which represents the variable.

For training, 150 images were used for each class and 50 for validation, obtaining an efficiency of 97%.

The CNN failed to classify the symbol "-" and the number "1". Fig. 15 shows the similarities between both characters after the cropping and resizing process.

We added a condition to solve the problem regarding the confusion of the symbol "-" and the number "1". If the width is larger than the height, the character is recognized as a minus sign. Otherwise, it is recognized as number one.

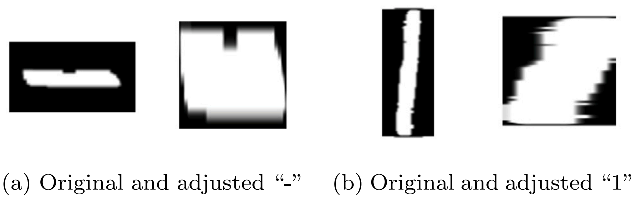


Fig. 15. Comparison between "-" and "1".

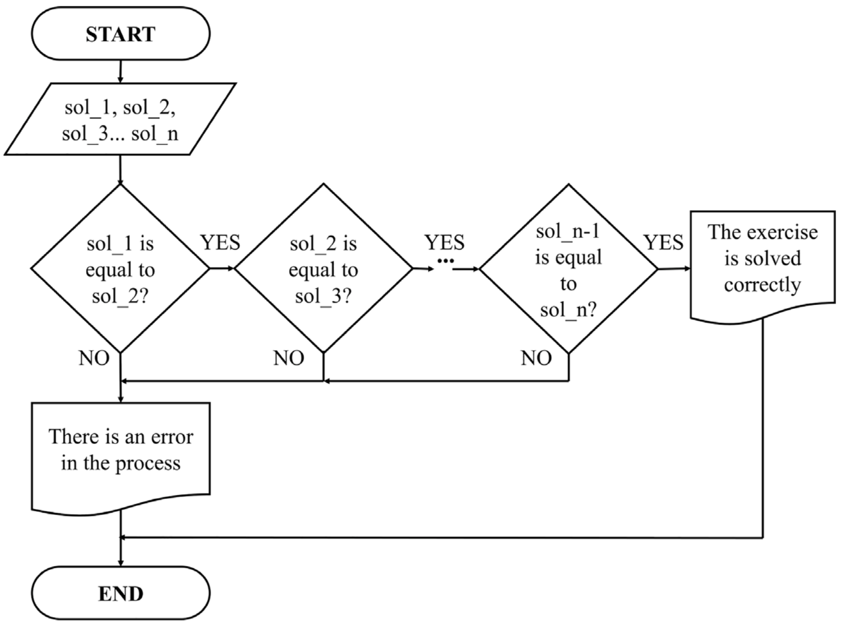

START
sol_1, sol_2, sol_3... sol_n
sol_1 is equal to sol_2?
YES
sol_2 is equal to sol_3?
YES
...
sol_n-1 is equal to sol_n?
YES
The exercise is solved correctly
NO
NO
NO
There is an error in the process
END

Fig. 16. Revision of the solving procedure of the linear equation.

4.5. *Revision of the solving procedure*

The solution of each equation is obtained and compared consecutively. The revision starts with the solution of the first equation. Fig. 16 shows the general flow diagram of the reviewing procedure.

The letter "n" represents the number of equations generated in Section 4.3.2 and will be the number of lines to review. For example, Fig. 17 shows an equation solved correctly and the reviewing procedure generated.

In Fig. 18, an incorrect resolution procedure is shown. The reviewing procedure stops after the first comparison since the error is located in the second line. Therefore, the solutions generated in the first and second equations are different, as described in Fig. 19.

The idea of comparing the solutions of the individual equations to review the procedure lies in the definition of equivalent equations. The equivalence of two equations means that they share the same solution set. A way to determine the equivalence of two equations is by transforming one equation into the other using valid transformations (Oleksik, 2019). The two rules derived from the Fundamental Axiom of Equations cited in Section 3 are examples of valid transformations.

The equations used in the experiments are linear equations with one variable. Consequently, the set of solutions of each equation is reduced to an element represented graphically by a single point on a straight line (or axis), as shown in Fig. 20.

Following the definition of equivalent equations, if two equations have the same solution, it is established that they are equivalent. Therefore, they are the same equation presented differently. Essentially, the procedure of solving an equation is exactly: presenting a final equation that is just a different and simpler version than the original.

5. Experiments and results

The experiments were performed using an HP laptop with an Intel Core i5 processor with 8 Gb of RAM. The graphical user interface (GUI) developed to facilitate the execution of the experiments is shown in Fig. 21.

The use of the interface consists of three steps: (i) insert the image of the solving procedure of a linear equation, (ii) introduce the number of individual equations in the solving procedure and start the review, and (iii) receive feedback.

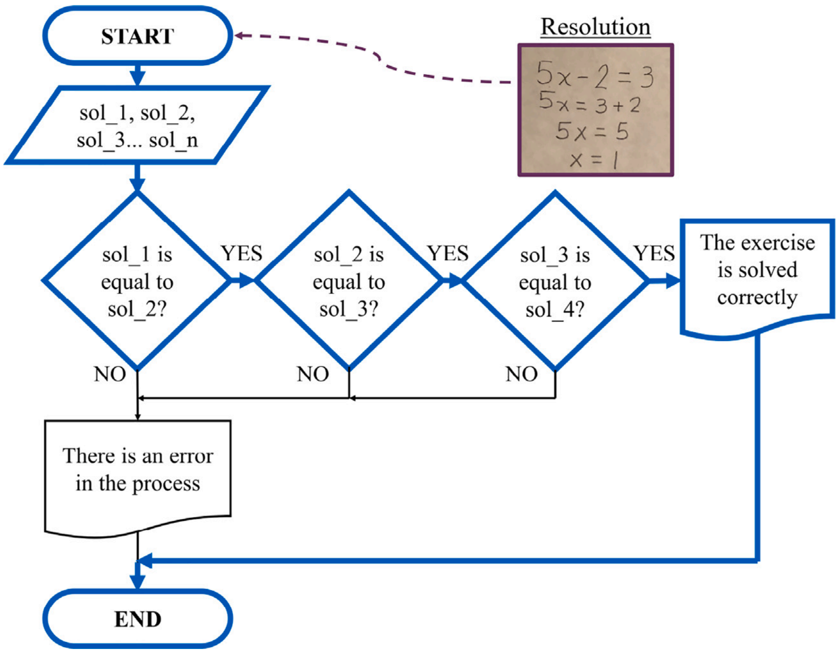


Fig. 17. The reviewing procedure of a correct resolution.

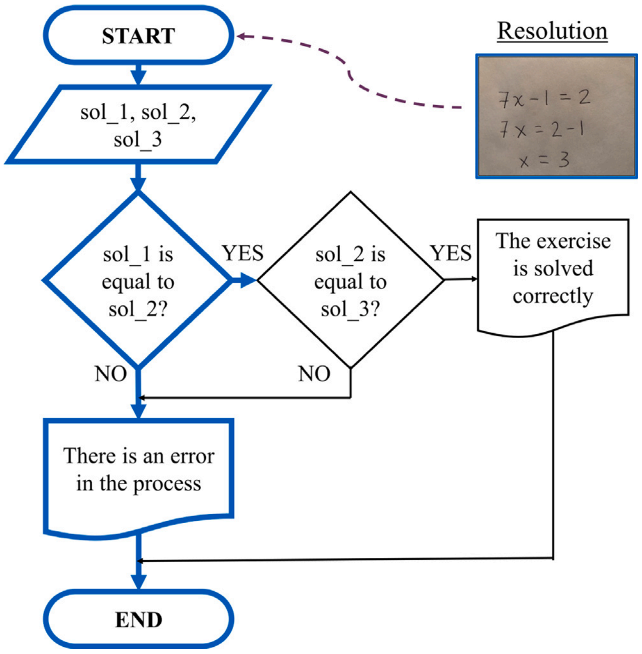
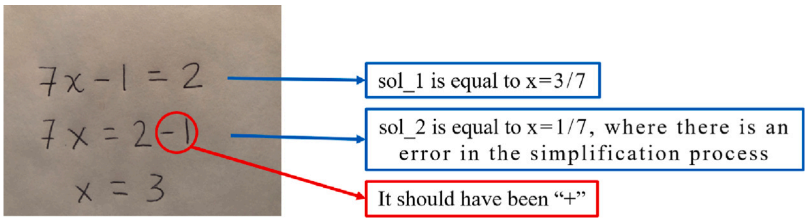


Fig. 18. The reviewing procedure of an incorrect resolution.

We ask fifteen undergraduates to solve a linear equation in one variable for the experiments. The participants wrote the equations on a white sheet of paper, and the acquisitions are shown in Fig. 22. Twelve students used pencils and the students that wrote equations 9, 12, and 14 used a black pen.

Each student used a mobile device to acquire the image. Therefore, different lighting conditions and image resolutions can be observed. Eleven equations were solved correctly (1–6, 8, 12–15), and four incorrectly (7, 9, 10, 11).

The resolution of each image in Fig. 22 is shown in Table 3.



sol_1 is equal to x=3/7

sol_2 is equal to x=1/7, where there is an error in the simplification process

It should have been "+"

Fig. 19. Description of the error corresponding to the incorrect resolution on Fig. 22(9).

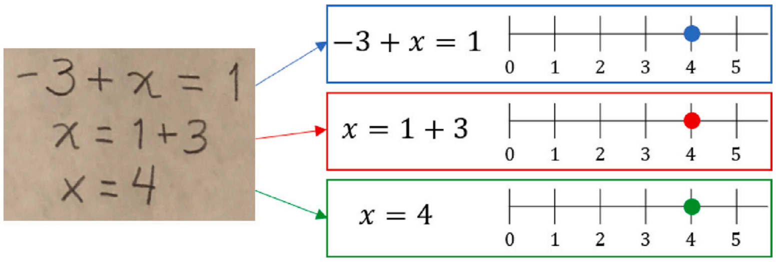


Fig. 20. Graphic representation of each equivalent equation.

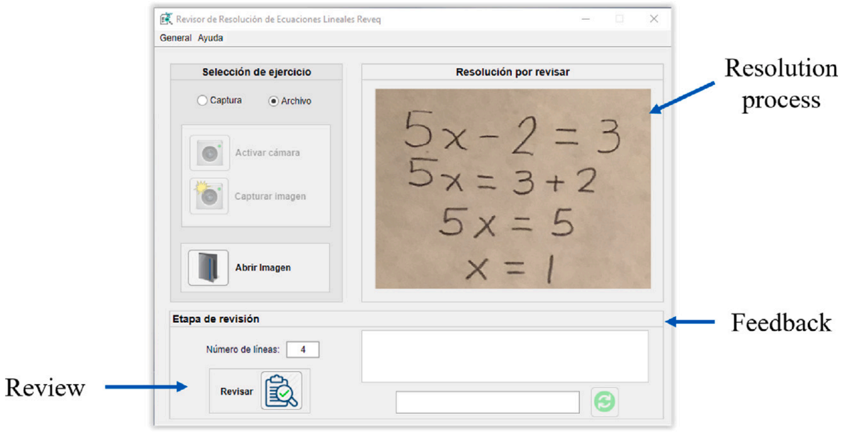


Fig. 21. GUI implemented for revising the solving procedure of a linear equation.

Table 3
Resolution of the 15 linear equations.

Equation	Area (pixels, px)	Equation	Area (pixels, px)
1	689 × 581 = 400 309	9	4032 × 3024 = 12 192 768
2	605 × 413 = 249 865	10	929 × 417 = 387 393
3	609 × 553 = 336 777	11	873 × 581 = 507 213
4	1177 × 620 = 729 740	12	599 × 467 = 279 733
5	821 × 557 = 457 297	13	720 × 1280 = 921 600
6	965 × 669 = 645 858	14	3022 × 2773 = 8 380 006
7	617 × 525 = 323 925	15	2600 × 1704 = 4 430 400
8	613 × 497 = 304 661		

The methodology proposed in Section 4 was applied to each of the 15 equations. An average of 20 runs (reviewing cycles) per revision were conducted to confirm that the results in each cycle were repeatable. The results obtained for character recognition and the solving procedure are presented in Table 4, where each column is described below.

A: Number of the equation from Fig. 22.
B: Total number of characters (each equal symbol was considered as two characters).
C: Number of characters segmented correctly.

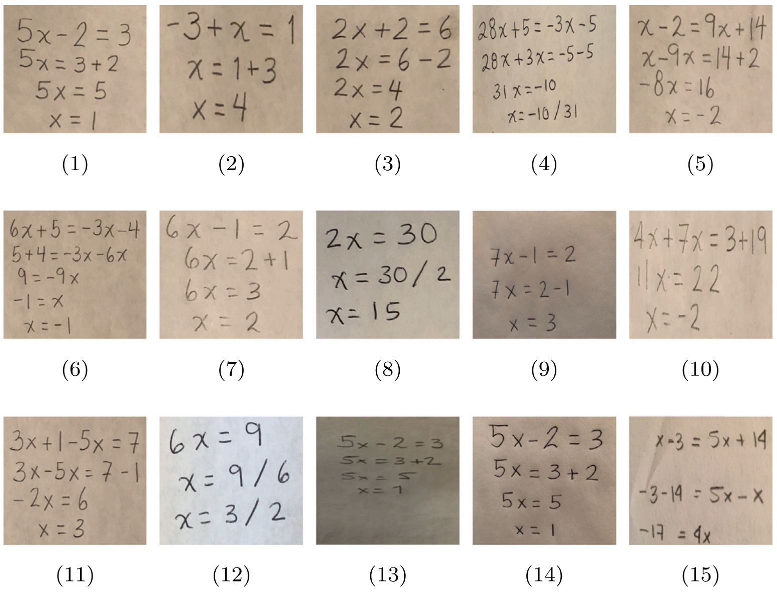


Fig. 22. Set of 15 linear equations used to conduct the experiments.

Table 4
Results obtained for the 15 linear equations.

A	B	C	D	E	F	G	H
1	23	23	23	4	4	No	Yes
2	17	17	16	3	1	No	No
3	23	23	23	4	4	No	Yes
4	41	41	41	4	4	No	Yes
5	32	32	32	4	4	No	Yes
6	38	38	38	5	5	No	Yes
7	23	23	23	4	4	Yes	Yes
8	18	18	18	3	3	No	Yes
9	18	18	18	3	3	Yes	Yes
10	23	23	23	3	3	Yes	Yes
11	30	30	30	4	4	Yes	Yes
12	17	17	17	3	3	No	Yes
13	23	23	22	4	3	No	No
14	23	23	23	4	4	No	Yes
15	28	28	28	3	3	No	Yes

D: Number of characters recognized correctly.
E: Total number of equations in the procedure.
F: Number of equations solved correctly.
G: Does the procedure contain any errors?
H: Was the procedure reviewed correctly?

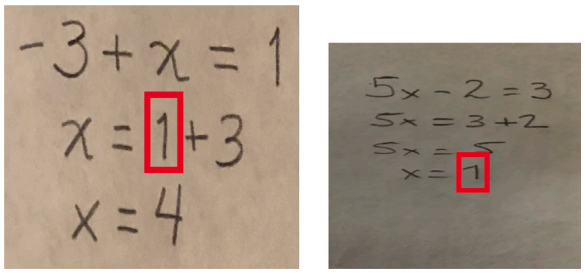
As can be seen from Table 4, some characters were wrongly recognized in equations 2 and 13 (column D). The number 1 was recognized as 7 in both cases (see Fig. 23). The character recognition results are shown in Table 5.

Due to the failure in the character recognition stage, the revision of the solving process for both equations was performed incorrectly. Therefore, an efficiency of 86.66% was obtained.

An option for correcting the wrongly recognized characters was implemented in the GUI. The user can manually fix the character and continue the procedure revision, as shown in Fig. 24. With the variant implemented, the revision stage can always be performed successfully. Therefore, there is a final efficiency of 100% in the revision stage. The disadvantage could be that the user manually solved the character recognition problem.

5.1. Invariance to transformations

We performed an additional set of tests to observe the robustness of our proposal. For this test, original images were modified by applying geometric transformations, changing the brightness, and adding noise. The explanation of the geometric transformations applied is shown in Table 6.

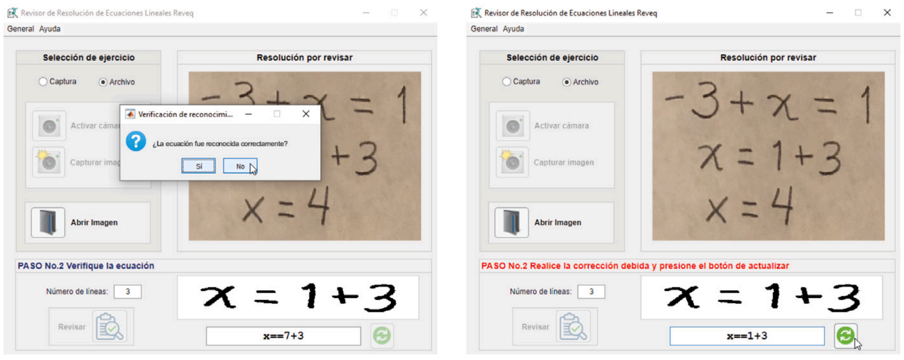


(a) Procedure 2 (b) Procedure 13

Fig. 23. Procedures with wrongly recognized characters.

Table 5
Results obtained in the character recognition phase.

Variables	Classes														Total
	0	1	2	3	4	5	6	7	8	9	+	−	/	×	
True positive	0	17	25	17	7	16	11	5	1	7	14	114	2	52	288
True negative	290	271	265	273	283	274	279	283	289	283	276	176	288	238	–
False positive	0	0	0	0	0	0	0	2	0	0	0	0	0	0	–
False negative	0	2	0	0	0	10	0	0	0	0	0	0	0	0	2
Real YES	0	19	25	17	7	16	11	5	1	7	14	114	2	52	290
Real NO	290	271	265	273	285	274	279	283	289	283	276	176	288	238	–
Predicted YES	0	17	25	17	7	16	11	7	1	7	14	114	2	52	–
Predicted NO	290	273	265	273	283	274	279	283	289	283	276	176	288	238	–
Variables	0	1	2	3	4	5	6	7	8	9	+	−	/	×	Total
Accuracy	1.00	0.99	1.00	1.00	1.00	1.00	1.00	0.99	1.00	1.00	1.00	0.99	1.00	1.00	0.99
Error rate	0.00	$6e^{-3}$	0.00	0.00	0.00	0.00	0.00	$6e^{-3}$	0.00	0.00	0.00	0.00	0.00	0.00	$1e^{-3}$
Sensitivity	1.00	0.89	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	1.00	1.00	0.99
Specificity	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	1.00	1.00	1.00	0.99	1.00	1.00	0.99
Precision	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	1.00	1.00	1.00	0.71	1.00	1.00	0.97



¿La ecuación fue reconocida correctamente?

PASO No.2 Verifique la ecuación

Número de líneas: 3

x==7+3

PASO No.2 Realice la corrección debida y presione el botón de actualizar

Número de líneas: 3

x==1+3

(a) Verification of character recognition (b) Character correction

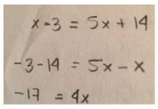
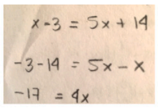
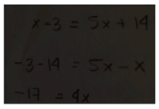
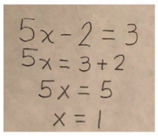
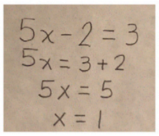
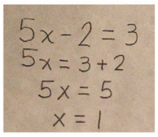
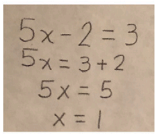
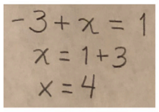
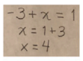
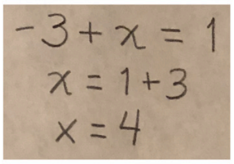
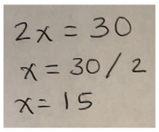
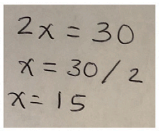
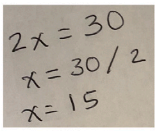
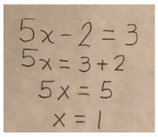
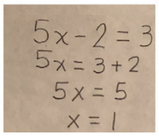
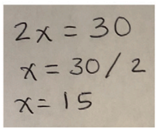
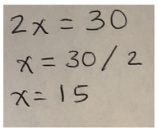
Fig. 24. Correction of the wrongly recognized characters.

(a) Brightness. Two filters were implemented to modify the brightness of the original images. **(1)** A white layer filter was applied. A glow around bright areas was added, which caused the image to become lighter. **(2)** A lighting effects filter was applied to delete the light source and darken the image.

(b) Noise. Three different noise types were added to the original images. **(1)** Salt & pepper (black and white pixels) noise was randomly added to the image. The noise was inserted to modify 1% of the total number of pixels. **(2)** RGB noise was applied to the image using a normal distribution. Therefore, slight noise is added to most pixels, while fewer pixels are affected by more extreme values. **(3)** A spread filter was applied to the image to swap each pixel with another randomly chosen pixel. The distance that pixels were moved along in the horizontal and vertical axes was 5.

(c) Scaling. The size of the original images was modified. **(1)** The image size was modified with a scale factor of 0.5 (halved). **(2)** The image size was modified with a scale factor of 1.5 (the size increased).

Table 6
Transformations applied to original images.

Transformation	**Transformations applied**			
a) Brightness	Original	**1:** Lighter	**2:** Darker	
				
b) Noise	Original	**1:** Salt & Pepper	**2:** RGB	**3:** Spread
				
c) Scaling	Original	**1:** Smaller	**2:** Larger	
				
d) Rotation	Original	**1:** Negative	**2:** Positive	
				
e) Translation	Original	**1:** Bottom right	Original	**2:** Top left
				

(d) Rotation. The orientation of the original images was changed. **(1)** The image was rotated 10° clockwise. **(2)** The image was rotated 10° counterclockwise.

(e) Translation. The position of the original image was modified. **(1)** The characters on the first image were moved to the bottom right of the image. **(2)** The characters on the second image were moved to the top left of the image.

The results obtained after performing the tests with transformed images were satisfactory for all stages of the proposed methodology (accuracy 100%). In other words, the segmentation, character recognition, and review of the solving process stages maintained their effectiveness throughout the tests performed with all the transformed images.

5.2. Discussion, implications and open challenges

The experiments were performed to analyze the behavior of the proposed methodology regarding character recognition and the revision of the solving procedure of linear equations. It should be noted from the results obtained that the system's accuracy regarding character recognition is 99%. Furthermore, the efficiency regarding the revision stage is 86.66%.

Based on the results obtained, the proposal could be an alternative tool to be used in the classroom. With our proposal, students can automatically review the process of solving a linear equation at any time. However, we will conduct a study to observe the real implications in the future. Students would be expected to be comfortable with the use of technology. Also, it will be essential to see the reaction of students who dislike using technology.

On the other hand, it will be desirable to observe if teachers are willing to use the proposal. The problem is that many teachers do not like to use new technologies and prefer to continue using the classic methods. However, we will try to convince them that using the proposal could reduce the time spent revising the exercises. In addition, it is expected that with this new proposal, more researchers will be encouraged to propose novel recommendations to revise the linear equation solving procedure.

Solving linear equations, and equations, in general, is a difficult task where students must face different challenges, such as:

- A wide variety of equation types.
- Different ways to solve an equation.

- Varied solution complexity depending on the type of equation.

Likewise, the revision of the solving procedure implies other types of challenges for teachers due to the different ways in which an equation can be solved. For example:

- Distinct fonts and writing styles.
- Different writing tools (pencils, pens, markers, colored pencils).
- Different types of paper.
- A variety of symbols used for arithmetical and algebraic operations.
- Different ways to perform an operation.

Based on the open challenges, we have established in Section 4.1 a set of rules to achieve the first prototype for reviewing the linear equation solving procedure which, as far as we know, has not been reported in the literature. However, we recognize that it is essential to be able to address more complex problems. Therefore, in the future, we aim to create a complete system that can review any mathematical expression.

A summary of the achievements obtained with our proposal is presented following.

- Correction of the problem of non-continuous strokes when scanning low- and high-resolution images.
- Denoising in low- and high-resolution images.
- Segmentation of the individual equations through the position of the segmented characters.
- Segmentation of individual characters.
- Recognition of individual characters corresponding to 14 different classes with an overall system accuracy greater than 90%.
- Differentiation of two similar characters through a comparison rule.
- Revision of the process of solving an equation through a comparison rule.
- Design of an application to review the solving process of an equation.
- Adaptation of the system to manually correct wrongly recognized characters through the application.

6. Conclusions and future work

This paper proposed a methodology for automatically revising the linear equations solving procedure with one unknown.

The results revealed that with the proposal, it is possible to recognize handwritten characters and effectively perform the automatic revision of the procedure of solving a linear equation with one unknown. It is important to note that although multiple systems solve linear equations, no systems have yet been reported that can automatically review the solving procedure of the equation.

Various elements were detected during the project's development that require future monitoring. Therefore, in future work, we propose improving the character classification problems presented during the experiments. Second, design a new filter to correct lighting problems in captured images. Third, create an algorithm to automatically detect the number of equations in a resolution procedure to avoid requesting the data from the user. Fourth, expand the image data set and the type of equations that the proposed system can review. Fifth, the evaluation of the proposed system with various test groups. Finally, the design of a mobile application of the proposed system.

CRediT authorship contribution statement

María Cristina Guevara Neri: Conceptualization, Methodology, Software, Writing – original draft, Writing – review & editing. **Osslan Osiris Vergara Villegas:** Conceptualization, Methodology, Formal analysis, Writing – original draft, Writing – review & editing. **Vianey Guadalupe Cruz Sánchez:** Conceptualization, Methodology, Formal analysis, Writing – original draft, Writing – review & editing. **Humberto de Jesús Ochoa Domínguez:** Methodology, Investigation, Visualization, Validation. **Manuel Nandayapa:** Methodology, Investigation, Writing – review & editing. **Juan Humberto Sossa Azuela:** Formal analysis, Investigation, Writing – review & editing.

Data availability

Data will be made available on request.

Acknowledgments

Mará Cristina Guevara thanks UACJ and CONACYT for support and scholarship to undertake her doctoral studies. Humberto Sossa thanks the Instituto Politécnico Nacional, Mexico and CONACYT, Mexico for the economic support under funds 20190007 and 65 (Fronteras de la Ciencia) to pursuit this research.

References

Abdullah, A., Tan, J., & Hu, M. (2021). A novel handwritten digit classification system based on convolutional neural network approach. *Sensors*, *21*, 1–26. http://dx.doi.org/10.3390/s21186273.

Aggarwal, R., Pandey, S., Kumar, A., & Harit, G. (2022). Survey of mathematical expression recognition for printed and handwritten documents. *IETE Technical Review*, *1*, 1–10. http://dx.doi.org/10.1080/02564602.2021.2008277.

Aghalari, M., Aghagolzadeh, A., & Ezoji, M. (2021). Brain tumor image segmentation via asymmetric/symmetric UNet based on two-pathway-residual blocks. *Biomedical Signal Processing and Control*, *69*(3), 1–11. http://dx.doi.org/10.1016/j.bspc.2021.102841.

Ahlawat, S., Choudhary, A., Nayyar, A., Singh, S., & Yoon, B. (2020). Improved handwritten digit recognition using convolutional neural networks (CNN). *Sensors*, *20*, 1–18. http://dx.doi.org/10.3390/s20123344.

Alabdulakareem, E., & Jamjoom, M. (2020). Computer-assisted learning for improving ADHD individuals' executive functions through gamified interventions: A review. *Entertainment Computing*, *33*, 1–8. http://dx.doi.org/10.1016/j.entcom.2020.100341.

Albahli, S., Alhassan, F., Albattah, W., & Ullah, R. (2020). Handwritten digit recognition: Hyperparameters-based analysis. *Applied Sciences*, *10*, 1–17. http://dx.doi.org/10.3390/app10175988.

Alghazo, J., Latif, G., Alzubaidi, L., & Elhassan, A. (2019). Multi-language handwritten digits recognition based on novel structural features. *Journal of Imaging Science and Technology*, *63*, 1–10. http://dx.doi.org/10.2352/J.ImagingSci.Technol.2019.63.2.020502.

Ali, S., Li, J., Pei, Y., Saqlain, M., Shaukat, Z., & Azeem, M. (2020). An effective and improved CNN-ELM classifier for handwritten digits recognition and classification. *Symmetry*, *12*, 1–15. http://dx.doi.org/10.3390/sym12101742.

Ali, S., Shaukat, Z., Azeem, M., Sakhawat, Z., Mahmood, T., & Rehman, k. (2019). An efficient and improved scheme for handwritten digit recognition based on convolutional neural network. *Springer Nature Applied Sciences*, *1*, 1–9. http://dx.doi.org/10.1007/s42452-019-1161-5.

Alvear, R., Sancho, J., & Figueiras, A. (2019). On improving CNNs performance: The case of MNIST. *Information Fusion*, *52*(1), 106–109. http://dx.doi.org/10.1016/j.inffus.2018.12.005.

Aly, S., & Almotairi, S. (2020). Deep convolutional self-organizing map network for robust handwritten digit recognition. *IEEE Access*, *8*, 107035–107045. http://dx.doi.org/10.1109/ACCESS.2020.3000829.

Anitei, D. (2020). *Development of a scalable database for recognition of printed mathematical expressions* (Ph.D. thesis), Universitat Politècnica de València.

Arias, F. (2022). *Método de reconocimiento de texto en imágenes naturales* (Ph.D. thesis), Universidad Autónoma de Ciudad Juárez.

Baldominos, A., Saez, Y., & Isasi, P. (2019). A survey of handwritten character recognition with MNIST and EMNIST. *Applied Sciences*, *9*, 1–16. http://dx.doi.org/10.3390/app9153169.

Baldor, A. (2019). *Algebra* (4th ed.). Larousse-Grupo Editorial Patria S.A. de C.V..

Boufenar, C., Kerboua, A., & Batouche, M. (2018). Investigation on deep learning for off-line handwritten arabic character recognition. *Cognitive Systems Research*, *50*, 180–195. http://dx.doi.org/10.1016/j.cogsys.2017.11.002.

Chan, C. (2020). Stroke extraction for offline handwritten mathematical expression recognition. *IEEE Access*, *8*, 61565–61575. http://dx.doi.org/10.1109/ACCESS.2020.2984627.

Coronado, A. (2017). The mathematics anxiety: A transcultural perspective. *Procedia - Social and Behavioral Sciences*, *237*, 1061–1065. http://dx.doi.org/10.1016/j.sbspro.2017.02.155.

Cullen, C., Hertel, J., & Nickels, M. (2020). The roles of technology in mathematics education. *The Educational Forum*, *84*, 166–178. http://dx.doi.org/10.1080/00131725.2020.1698683.

Dash, K., Puhan, N., & Panda, G. (2018). Unconstrained handwritten digit recognition using perceptual shape primitives. *Pattern Analysis & Applications*, *21*, 413–436. http://dx.doi.org/10.1007/s10044-016-0586-3.

Duc, A., Dai, H., Indurkhya, B., & Nakagawa, M. (2019). Stroke order normalization for improving recognition of online handwritten mathematical expressions. *International Journal on Document Analysis and Recognition (IJDAR)*, *22*, 29–39. http://dx.doi.org/10.1007/s10032-019-00315-2.

Duc, A., Indurkhya, B., & Nakagawa, M. (2019). Pattern generation strategies for improving recognition of handwritten mathematical expressions. *Pattern Recognition Letters*, *128*, 255–262. http://dx.doi.org/10.1016/j.patrec.2019.09.002.

Guevara, M. (2021). *Metodología para la revisión automática del proceso de resolución de ecuaciones lineales con una incógnita* (Ph.D. thesis), Universidad Autónoma de Ciudad Juárez.

Guimaraes, B., Firmino, J., Tsisar, S., Viana, B., Pinto, M., Vieira, P., Cruz, R., & Ferreira, A. (2019). The role of anatomy computer-assisted learning on spatial abilities of medical students. *Anatomical Sciences Education*, *12*(2), 138–153. http://dx.doi.org/10.1002/ase.1795.

He, F., Tan, J., & Bi, N. (2020). Handwritten mathematical expression recognition: A survey. In *Proc. of the international conference on pattern recognition and artificial intelligence (ICPRAI)* (pp. 55–66). Springer.

Huang, J., Tan, J., & Bi, N. (2020). Overview of mathematical expression recognition. In *Proc. of the international conference on pattern recognition and artificial intelligence (ICPRAI)* (pp. 41–54). Springer.

Jha, G., & Cecotti, H. (2020). Data augmentation for handwritten digit recognition using generative adversarial networks. *Multimedia Tools and Applications*, *79*, 35055–35068. http://dx.doi.org/10.1007/s11042-020-08883-w.

Kukreja, V. (2021). A retrospective study on handwritten mathematical symbols and expressions: Classification and recognition. *Engineering Applications of Artificial Intelligence*, *103*, 1–32. http://dx.doi.org/10.1016/j.engappai.2021.104292.

Kukreja, V., & Sakshi (2022). Machine learning models for mathematical symbol recognition: A stem to stern literature analysis. *Multimedia Tools and Applications*, *81*, 28651–28687. http://dx.doi.org/10.1007/s11042-022-12644-2.

Kulkarni, S., & Rajendran, B. (2018). Spiking neural networks for handwritten digit recognition—Supervised learning and network optimization. *Neural Networks*, *103*, 118–127. http://dx.doi.org/10.1016/j.neunet.2018.03.019.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324. http://dx.doi.org/10.1109/5.726791.

LeCun, Y., Cortes, C., & Burges, C. (2018). The MNIST database. URL: http://yann.lecun.com/exdb/mnist/.

Lin, Y., Wang, C., & Zeng, J. (2017). A case study on mathematical expression recognition to GPU. *The Journal of Supercomputing*, *73*(8), 3333–3343. http://dx.doi.org/10.1007/s11227-016-1819-3.

Lu, Y., Qin, X., Fan, H., Lai, T., & Li, Z. (2021). WBC-Net: A white blood cell segmentation network based on UNet++ and ResNet. *Applied Soft Computing*, *101*(3), 1–11. http://dx.doi.org/10.1016/j.asoc.2020.107006.

Madakannu, A., & Selvaraj, A. (2020). DIGI-Net: A deep convolutional neural network for multi-format digit recognition. *Neural Computing and Applications*, *32*, 11373–11383. http://dx.doi.org/10.1007/s00521-019-04632-9.

Meeter, M. (2021). Primary school mathematics during the COVID-19 pandemic: No evidence of learning gaps in adaptive practicing results. *Trends in Neuroscience and Education*, *25*(1), 1–8. http://dx.doi.org/10.1016/j.tine.2021.100163.

Mei, B., Brown, G., & Teo, T. (2018). Toward an understanding of preservice english as a foreign language teachers' acceptance of computer-assisted language learning 2.0 in the people's republic of China. *Journal of Educational Computing Research*, *56*(1), 1–31. http://dx.doi.org/10.1177/0735633117700144.

Min, C., Bi, C., Guo, Z., Kang, L., & Ping, C. (2020). An adaptive fractional-order BP neural network based on extremal optimization for handwritten digits recognition. *Neurocomputing*, *391*, 260–272. http://dx.doi.org/10.1016/j.neucom.2018.10.090.

Minh, K., Duc, A., Indurkhya, B., & Nakagawa, M. (2018). Augmented incremental recognition of online handwritten mathematical expressions. *International Journal on Document Analysis and Recognition (IJDAR)*, *21*, 253–268. http://dx.doi.org/10.1109/TMM.2018.2844689.

Oleksik, N. (2019). Transforming equations equivalently? – theoretical considerations of equivalent transformations of equations. In *Proc. of the eleventh congress of the European society for research in mathematics education* (pp. 1–10). Utrecht University.

Pal, A., & Singh, K. (2022). R-GRU: Regularized gated recurrent unit for handwritten mathematical expression recognition. *Multimedia Tools and Applications*, *1*, 1–15. http://dx.doi.org/10.1007/s11042-022-12889-x.

Phangtriastu, M., Harefa, J., & Felita, D. (2017). Comparison between neural network and support vector machine in optical character recognition. *Procedia Computer Science*, *116*, 351–357. http://dx.doi.org/10.1016/j.procs.2017.10.061.

Ptucha, R., Petroski, F., Pillai, S., Brockler, F., Singh, V., & Hutkowski, P. (2019). Intelligent character recognition using fully convolutional neural networks. *Pattern Recognition*, *88*, 604–613. http://dx.doi.org/10.1016/j.patcog.2018.12.017.

Qiao, J., Wang, G., Li, W., & Chen, M. (2018). An adaptive deep Q-learning strategy for handwritten digit recognition. *Neural Networks*, *107*, 61–71. http://dx.doi.org/10.1016/j.neunet.2018.02.010.

Sakshi, & Kukreja, V. (2022a). A hybrid SVC-cnn based classification model for handwritten mathematical expressions (numbers and operators). In *Proc. of the 2022 international conference on decision aid sciences and applications (DASA)* (pp. 321–325). IEEE.

Sakshi, & Kukreja, V. (2022b). Segmentation and contour detection for handwritten mathematical expressions using openCV. In *Proc. of the 2022 international conference on decision aid sciences and applications (DASA)* (pp. 305–310). IEEE.

Saqib, N., Foysal, K., Prasanth, V., & Abdelgawad, A. (2022). Convolutional-neural-network-based handwritten character recognition: An approach with massive multisource data. *Algorithms*, *15*, 1–25. http://dx.doi.org/10.3390/a1504012910.1002/tee.23278.

Shinde, S., Alagirisamy, M., Bhalke, D., & Wadhwa, L. (2021). Complex mathematical expressions recognition using support vector machine as a classifier. *Information Technology in Industry*, *9*(3), 584–588.

Shuvo, S., Hasan, F., Ahmed, M., Hossain, S., & Abujar, S. (2021). MathNET: Using CNN bangla handwritten digit, mathematical symbols, and trigonometric function recognition. *Soft Computing Techniques and Applications*, *1248*, 515–523. http://dx.doi.org/10.1007/978-981-15-7394-1_47.

Thurm, D., & Barzel, B. (2022). Teaching mathematics with technology: A multidimensional analysis of teacher beliefs. *Educational Studies in Mathematics*, *109*, 41–63. http://dx.doi.org/10.1007/s10649-021-10072-x.

Tokac, U., Novak, E., & Thompson, C. (2019). Effects of game-based learning on students' mathematics achievement: A meta-analysis. *Journal of Computer Assisted Learning*, *35*(3), 407–420. http://dx.doi.org/10.1111/jcal.12347.

Urazoe, K., Kuroki, N., Hirose, T., & Numa, M. (2021). Combination of convolutional neural network architecture and its learning method for rotation-invariant handwritten digit recognition. *IEEJ Transactions on Electrical and Electronic Engineering*, *16*, 161–163. http://dx.doi.org/10.1002/tee.23278.

Verbruggen, S., Depaepe, F., & Torbeyns, J. (2021). Effectiveness of educational technology in early mathematics education: A systematic literature review. *International Journal of Child-Computer Interaction*, *27*, 1–42. http://dx.doi.org/10.1016/j.ijcci.2020.100220.

Wang, J., Du, J., Zhang, J., Wang, B., & Ren, B. (2021). Stroke constrained attention network for online handwritten mathematical expression recognition. *Pattern Recognition*, *119*, 1–15. http://dx.doi.org/10.1016/j.patcog.2021.108047.

Weng, Y., & Xia, C. (2020). A new deep learning-based handwritten character recognition system on mobile computing devices. *Mobile Networks and Applications*, *25*, 402–411. http://dx.doi.org/10.1007/s11036-019-01243-5.

Wook, M., Pedrotty, D., & Bryant, B. (2020). Effects of computer-assisted instruction on the mathematics performance of students with learning disabilities: A synthesis of the research. *Exceptionality*, *28*(1), 1–15. http://dx.doi.org/10.1080/09362835.2019.1579723.

Wu, J., Yin, F., Zhang, Y., Zhang, X., & liu, C. (2020). Handwritten mathematical expression recognition via paired adversarial learning. *International Journal of Computer Vision*, *128*, 2386–2401. http://dx.doi.org/10.1007/s11263-020-01291-5.

Yu, Z., Nisha, S., Yazid, H., & Aziz, M. (2021). Performance analysis of otsu thresholding for sign language segmentation. *Multimedia Tools and Applications*, *80*, 21499–21520. http://dx.doi.org/10.1007/s11042-021-10688-4.

Zhang, J., Du, J., & Dai, L. (2019). Track, attend, and parse (TAP): An end-to-end framework for online handwritten mathematical expression recognition. *IEEE Transactions on Multimedia*, *21*, 221–233. http://dx.doi.org/10.1109/TMM.2018.2844689.

Zhang, J., Du, j., Yang, Y., Song, Y., & Dai, L. (2021). SRD: A tree structure based decoder for online handwritten mathematical expression recognition. *IEEE Transactions on Multimedia*, *23*, 2471–2480. http://dx.doi.org/10.1109/TMM.2020.3011316.

Zhang, J., Du, J., Zhang, S., Liu, D., Hu, Y., Hu, J., Wei, S., & Dai, L. (2017). Watch, attend and parse: An end-to-end neural network based approach to handwritten mathematical expression recognition. *Pattern Recognition*, *71*, 196–206. http://dx.doi.org/10.1016/j.patcog.2017.06.017.

Zhang, T., Moucheré, H., & Viard, C. (2020). A tree-BLSTM-based recognition system for online handwritten mathematical expressions. *Neural Computing Applications*, *32*, 4689–4708. http://dx.doi.org/10.1007/s00521-018-3817-2.

Zhang, D., Wang, L., Zhang, L., Tian, B., & Tao, H. (2020). The gap of semantic parsing: A survey on automatic math word problem solvers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *42*(9), 2287–2305. http://dx.doi.org/10.1109/TPAMI.2019.2914054.

Zhao, H., & Liu, H. (2020). Multiple classifiers fusion and CNN feature extraction for handwritten digits recognition. *Granular Computing*, *5*, 411–418. http://dx.doi.org/10.1007/s41066-019-00158-6.

Zhelezniakov, D., Zaytsev, V., & Radyvonenko, O. (2021). Online handwritten mathematical expression recognition and applications: A survey. *IEEE Access*, *9*, 38352–38373. http://dx.doi.org/10.1109/ACCESS.2021.3063413.