



Object Detection Based Software System for Automatic Evaluation of *Cursogramas* Images

Pablo Pytel^(✉), Matías Almad, Rocío Leguizamón, Cinthia Vegega^(✉),
and Ma Florencia Pollo-Cattaneo^(✉)

Grupo de Estudio en Metodologías de Ingeniería en Software (GEMIS), Facultad Regional
Buenos Aires, Universidad Tecnológica Nacional, Buenos Aires, Argentina
{ppytel, cvegega, fpollo}@frba.utn.edu.ar

Abstract. The aim of this work is to describe the tasks performed to carry out the development of a software system capable of detecting and recognizing the symbols of *Cursogramas* in images by using a Deep Learning model that has been trained from scratch. In this way, we seek to assist teachers of an undergraduate subject to automatically evaluate diagrams made as part of the practical exercise of their students. For this purpose, in addition to having carried out a process of understanding the problem and identifying the available data, tasks of technology selection and construction of each of the components that are part of the system are also carried out. Therefore, although the problem domain belongs to the field of university education, this work is more related to the engineering and technological aspect of the application of Artificial Intelligence to solve complex problems.

Keywords: *Cursogramas* · Object detection · Deep learning · Artificial intelligence

1 Introduction

A *Cursograma* is a work tool, which allows to represent graphically the movement of documents that correspond to a certain administrative procedure [1]. Its main objective is to be able to represent a routine, without falling into the complexity of the graph, since this can lead to misinterpretation [2]. Given its usefulness, this method is taught within the subject ‘Sistemas y Organizaciones’ in the first level of the career ‘Information Systems Engineering’ [3] within the Facultad Regional Buenos Aires of Universidad Tecnológica Nacional (Argentina). This annual course is part of the integrating core and crosses the curriculum at different levels [4].

As it usually happens with any type of practical exercise, in order to achieve a good handling of these diagrams, students must perform a large number of exercises. But, they also need to have feedback on the correction of the exercises. Since there are no automatic methods that allow revision, the evaluation work is carried out manually by teachers and assistants. In many cases, the proposal provided by the students may

present some minor differences. For this reason, the correction of each diagram requires a considerable amount of time, since the resolution must be analyzed in depth, in addition to paying special attention to the analysis process that the students carry out at the time of the resolution.

In this context, the objective of this work is to describe the tasks carried out for the implementation of a software system that allows the automatic evaluation of *Cursograma* diagrams made by students in the practical work of the subject. Such a tool would generate benefits for both teachers and students. For this purpose, first in Sect. 2 the description of the problem is presented with its main particularities that must be considered to carry out the implementation of the software system. Then, in Sect. 3, the proposed solution is presented describing each of the main components that have been necessary to develop in order to achieve the objective. The results obtained are shown in Sect. 4. Finally, Sect. 5 presents the conclusions and future lines of work.

2 Problem Description

The subject ‘Sistemas y Organizaciones’ belongs to the first year of the career and is therefore mandatory for students who have passed the entrance course to the career. In the year 2020 the number of new students has been approximately 1400, so there are courses with more than 80 students. This has motivated the subject chair to apply Artificial Intelligence technologies to assist the teaching-learning process [5]. Among the objectives of this Intelligent System is a software system in charge of reviewing, correcting and automatically evaluating the *Cursograma* diagrams made by the students as part of the practical exercise of the course.

As a result of several requirements elicitation sessions, applying an engineering process similar to the one proposed in [6], it has been possible to obtain the necessary information to determine the requirements and constraints to be considered in the development of this software system.

On the one hand, the symbols used in *Cursogramas* have been surveyed and can be found in [7]. Later, an “expert” of the chair has also been trained to know the main rules that define the valid ways in which the symbols should be connected to each other.

On the other hand, the main functionality is identified as the automatic evaluation of images that include the *Cursogramas* made by the students. For this purpose, not only the symbol template with the rules previously mentioned must be taken into account, but also the particularities of the statement of the exercise solved by the students. Likewise, the operation to be included in the system is defined, from the reception of the images from the students to the return with their revision. This review should provide the student with a quantitative evaluation of the exercise as well as a set of qualitative observations on corrections. This means that, for each image reviewed, the mistakes and problems encountered should be marked, indicating, in each case, the nature of the error.

Once the problem to be solved has been determined, a bibliographic search has been carried out to try to find similar developments that could be used or adapted to meet the elicited requirements. In this sense, several proposals have been found where Artificial Intelligence is used for the automatic evaluation of diagrams both in the field of education, as well as, for professional usage. However, the proposals found are oriented towards

UML-type diagrams [8–12] Engineering CAD Drawing [13], and Digital Logic Circuit diagrams [14] among others, which have different characteristics and symbols from *Cursogramas* and then are not useful for the considered problem. Consequently, it has been decided to implement a totally new software system oriented to the particularities of the subject in question.

3 Proposed Solution

Considering all the requirements and restrictions that the software system must fulfill, work has begun on its design and development. In order to fulfill the objective of the software system to be developed, it is essential to apply a technology that automatically allows recognizing the symbols presented in the *Cursograma* image, and also their relative location in the diagram in order to determine how they are connected to each other. The construction of this Detection Model is described in Sect. 3.1. Having achieved a Symbol Detection Model, then the development of the main component is started. This component deals with the evaluation of the *Cursograma* exercises solved by the students for a specific exercise. The details about the functionalities of this Evaluation program are presented in Sect. 3.2.

3.1 Construction of the Symbol Detection Model

Since it is preferred to use a ‘Machine Learning’ algorithm [15] that can “learn” to recognize the symbols, the application of Deep Learning Neural Networks [16], which are better known as ‘Deep Learning’, is selected. From the great variety of existing models for Deep Learning networks [17], a variety of Convolutional Neural Networks or CNN [18, 19] called ‘Object Detection Model’ [20] has been selected for this work. Since there are a large number of possible architectures for implementing Object Detection Models, specifically, three types are considered, the Faster R-CNN Inception version 2 [21], the SSD MobileNet version 2 [22] and the R-FCN ResNet 101 [23]. Although models of these three architectures are available that already “know” how to detect objects in an image, none are useful for working with *Cursograma* symbols. Therefore, only zero-trained architectures can be used by applying the corresponding algorithm to determine the set of values for a set of parameters in a way that represents the behavior of a set of available data [15, 29, 30].

However, in order to learn to detect the symbols, this training algorithm requires that, a set of additional “annotations”. In our case, for each image of a *Cursograma* we need an XML file [24] which indicates the coordinates of a region (or “box”) where each symbol is located and a “label” indicating the type of symbol in question. Since this information is not available (and its manual generation requires a lot of effort), it has been decided to develop an ad-hoc program to automatically generate random cases of *Cursogramas*. For each case, the diagram will be recorded in a PNG type image, and all the complementary information in the XML file. In order to carry out this case generation, the standardized format of symbols and rules defined by an “expert teacher” will be used.

Once this program has been developed, it is used to generate the cases (each one consisting of a PNG image and an associated XML file) for the construction of the model for the detection of the *Cursograma* diagram symbols. In the first instance, 1,000 cases are generated, 90% of which are used for training and the rest for validation. With them, the three types of architectures mentioned above are trained and validated. As can be seen in Table 1, none of them generate acceptable results. The SSD architecture has the worst results, while Faster R-CNN has the best results although it tends to detect fewer symbols present in the image than R-FCN so its recall is lower.

Since the processing speed is not as important as the model accuracy, it is decided to discard the SSD architecture and continue the training with the other two architectures by adding more examples. After adding 4,000 new cases (thus generating a total of 5,000), the architectures are re-trained and validated, obtaining the results shown in Table 1. As can be seen, now R-FCN is the one with the best results, being acceptable. After analyzing in detail, it is discovered that most of the errors have to do with the detection of the symbol that corresponds to a ‘Decision’. Although the ‘Decision’ symbol is quite a very simple symbol to recognize, two connections (generally one vertical and one horizontal) come out of this symbol, which seems to confuse the model. Therefore, it is decided to add another 500 new examples of diagrams where decisions appear in order to retrain the R-FCN architecture. As can be seen in the last row of Table 1, an almost perfect model is obtained.

Table 1. Results of the symbol detection model’s first training and validation.

Cases generated	Model architecture	Validation metrics		
		Accuracy	Precision	Recall
1,000	SSD	51.6%	74.1%	63.1%
	Faster R-CNN	63.7%	88.5%	69.4%
	R-FCN	56.3%	63.7%	82.9%
5,000	Faster R-CNN	59.7%	90.2%	63.9%
	R-FCN	99.5%	99.6%	99.9%
5,500	R-FCN	99.9%	99.9%	100%

Nevertheless, after a meeting where the expert teachers of the subject review the first results, they complained that that all the symbols always have the same letters and numbers, for example the documents have the same type (“F0”), while the controls and operations appear with “1”. Another shortcoming that they also consider important is the lack of examples of document copies, where each copy may have a different circuit. All these issues have more to do with the diagram generator program, and therefore a new version is developed that corrects them.

As the previously trained model is used to handling only symbols with “hardcoded” letters and numbers, a new one must be train to handle these changes using a new batch of generated cases (each one including a PNG and XML files). The new batch includes 11,934 cases for training and 1,326 cases for validation.

Despite of using the same architecture R-FCN, in order to obtain a reliable model, several training sessions have been performed. In each of these sessions, several training parameters has been tried to change to improve the model performance. Based on a later result analysis it is detected that the critical one has been the quantity of training steps. As it can be seen in Table 2, by increasing the quantity of cycles performed during the training, the model accuracy and recall change considerably. However, after the 50,000 steps milestone, the model performance starts again to decline because the model is over-trained and thus loses its generalization capabilities. On that account, the 50,000 training steps is selected to be used for the evaluation program. Although, its performance is affected by ‘false positives’, they are related to symbol connectors so this is not considered a problem to carry out the evaluation of the diagrams, and the construction of the Symbol Detection Model is considered as successfully completed.

Table 2. Results of the symbol detection model’s second training and validation.

Model architecture	Training steps	Validation metrics		
		Accuracy	Precision	Recall
R-FCN	10,000	97.5%	99.9%	97.5%
	20,000	98.5%	99.9%	98.5%
	50,000	99.5%	99.9%	99.5%
	100,000	99.2%	99.9%	99.2%

3.2 Implementation of the Evaluation Program

The Evaluation program is the software component that deals with the evaluation of the *Cursograma* exercises solved by the students. This is performed using another diagram that is used as a reference at the time of the review. This other diagram must obviously comply with both the general rules mentioned above, as well as the particularities of the exercise statement.

These two diagrams, the one provided by the students to be evaluated and the one provided by the teachers for references, are provided in the form of PNG images. Therefore, to carry out the evaluation, first the symbol Detection Model is applied on each image obtaining a list of detected symbols. Also, when a detected symbol includes a letter and/or number (e.g. a document), they are recognized using the open source Tesseract Optical Character Recognition (OCR) Engine [25, 28]. As a result, two lists are obtained (one for the students’ image and another for teacher’s) that for each detected symbol includes: the symbol class, its relative position in the diagram and the OCR string with the corresponding letters and numbers.

When the lists are available, they are processed by comparing them using an ad-hoc algorithm. For each the students’ symbol the following rules are analyzed:

- If there is a match in the teacher’s list where a symbol has the *same class, relative position and OCR string*, then the students’ symbol is marked as *correct*.

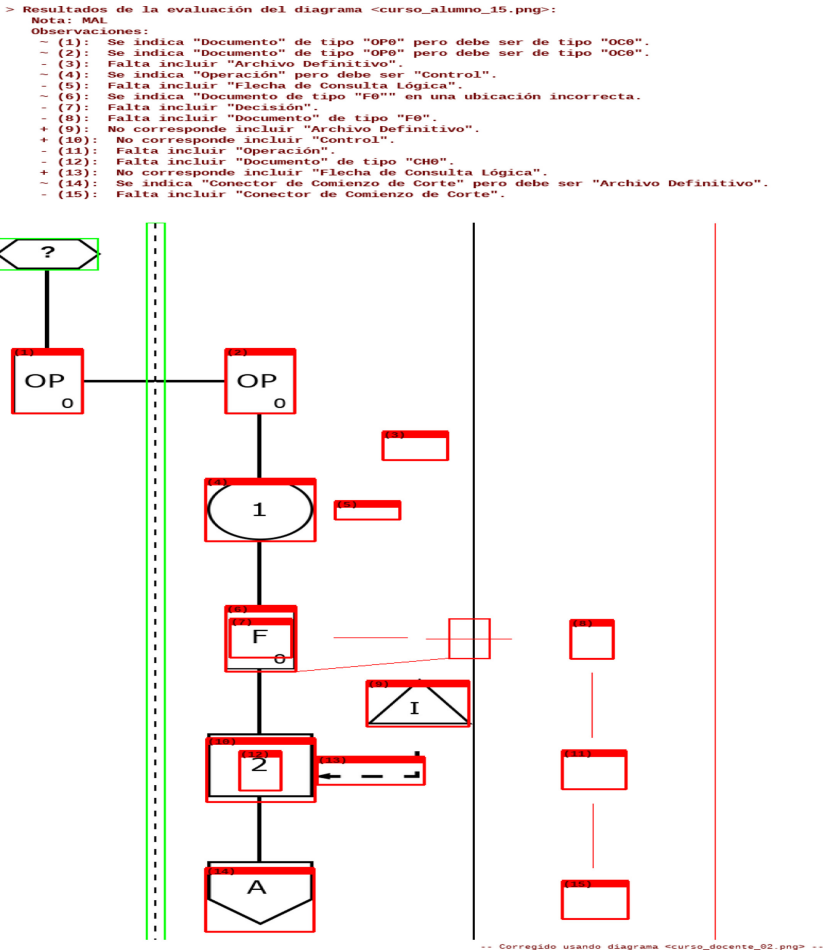


Fig. 1. Example of evaluating a students' diagram. (Color figure online)

- If there is a match in the teacher's list where a symbol has the *same class and relative position but different OCR string*, then the students' symbol is marked as *mistake in symbol text*.
- If there is a match in the teacher's list where a symbol has the *same class and OCR string but different relative position*, then the students' symbol is marked as *mistake in symbol location*.
- If there is a match in the teacher's list where a symbol has *different class but the same relative position and OCR string*, then the students' symbol is marked as *mistake in symbol class*.
- If there is not a match in the teacher's list, then the students' symbol is marked as *mistake of over indicated symbol*.

Finally, when the all the entries in the students' list are processed, if there is a teacher's symbol that *has not been matched*, then this teacher's symbol is marked as *mistake of missing symbol*.

The results of this evaluation algorithm are highlighted in the students' image marking them with colors (green color if it is correct, red color otherwise) as it can be seen in Fig. 1. In case of "missing" teacher's symbols, the location is established by a special function that uses position of other symbols matched in both diagrams as landmarks. Furthermore, a grade for the exercise is "calculated" and recorded in the header of the image with the corresponding observations where the detected mistakes are explained.

4 Results

This section presents the results of the tests performed to confirm that the developed Evaluation program works correctly. This is achieved by running the component on many different diagrams to review. In this case, it has been decided to use again completely randomly generated cases as explained in Sect. 4.1. This means that none of the test cases used here correspond to real exercises of the subject but, anyway, they are considered useful to check how the software behaves with different situations. Using these generated cases, in Sect. 4.2 a study case is utilize to explain the results of evaluating a students' diagram. Later, in Sect. 4.3 its robustness is verified by contrasting the results of normal and "noisy" images. Finally, in Sect. 4.4, the evaluation results are analyzed by expert professors of the subject to decide if the diagrams have been evaluated correctly.

4.1 Generation of Test Cases

In order to generate these test cases that are available in the repository [26], the following procedure used is as follows:

First, 5 different diagrams are generated, each with a different level of complexity, which are taken as reference images provided by the teacher. That is, these images (that stored in the "Teacher_diagrams" folder of [26]) are taken as if they were the correct resolution of an exercise and are used as a basis for the revision of the students' diagrams.

On the other hand, the students' diagrams are also generated automatically to obtain 10 new images. To these images a copy of the 5 reference images are added in order to be able to corroborate that the software works correctly in cases where there is no error. Moreover, a noise function is applied to these 15 images to generate images that will be used to verify the capabilities of the detection model. This function applies a kind of 'Salt-and-Pepper' noise [27] that changes the value of random pixels in order to generate unanticipated disturbances in the image. These changes are few but they are very noisy. As a result, the effect is similar to sprinkling white and black dots on the image. All the "noisy" images have the suffix "DAn" in their file name so they can be easily identified. In this way, a total of 30 students' exercises to evaluate are obtained and store in the "Students_diagrams" folder of [26].

Once all the test images have been generated, the automatic evaluation program is run for each combination, so that each of the 30 students' images is reviewed against each of the 5 reference teacher's images. Upon completing the evaluation of all combinations, it

is observed that the evaluation program takes between 5 and 8 s to process each image. In total then 150 evaluations are carried out and recorded in the “Evaluation_results” folder of [26]. Note that to facilitate their organization and search, in the name of these images the identifier of the teacher’s reference image is indicated first and then the image corresponding to the student’s exercise. Thus, for example, in Fig. 1, the results of evaluating the “noisy” student’s image ‘02’ by using as reference the teacher’s image ‘01’.

Finally, as the observations written in the evaluation image’s header are in spanish, to assist non-spanish speakers understand the detected mistakes, a multiple color version of each evaluation is generated. To highlight the different evaluation results, the color **Green** is used to identify the symbols that are correct, **Orange** to show a mistake of class, location or OCR string, **Violet** for those that are over indicated and **Red** for those that are missing. This version of the example shown previously in Fig. 1 can be seen below in Fig. 2. Also, the multiple is available for all the images in the “Evaluation_results” folder of [26] and are identified with the suffix “-mc” in their file name.

4.2 Analysis of a Case Study

The objective of the case study is demonstrating the results of evaluating a students’ diagram by analyzing the example shown in Fig. 1 and 2. This example is generated after evaluating the students’ diagram ‘15’ (shown in the left column of Table 3) based on the teacher’s reference diagram ‘02’ (shown in the right column of the same table).

As it can be noted, in spite of having a similar beginning, after the third symbol the students’ and teacher’s diagrams have a lot of differences. Therefore, it is clear why that the evaluation result image has a bad grade (“MAL” in spanish) and a lot of symbols highlighted in non-green color. But, in fact, this example is interesting because it includes all the types of evaluation mistakes, and that is why it is used as a case study.

In order to analyze the evaluation results, below each one of highlighted marks of Fig. 2 from top to bottom is explained. Also, for the purpose of facilitating this explanation, the Table 4 is presented where Fig. 2 is separated into parts and matched to the corresponding parts of the teacher’s reference image.

- 1) At the beginning of both diagrams, the first symbol is a “not identified process”, then it is a perfect match and the symbol is highlighted as *correct* in **green** color.
- 2) This “not identified process” is in an external sector of the organization, which is separated from the internal sector by a dotted line that is drawn in both diagrams, so it is also a perfect match and the symbol is highlighted as *correct* in **green** color.
- 3) As a result of this “not identified process”, a document is generated that is transferred to an internal sector of the organization. Despite that this transfer is carried out correctly, the type of document indicated (which is recognized by the OCR engine) is incorrect. It can be noted that the teacher’s documents are an “OC-0” but the students’ are an “OP-0”. As a result, the *mistake in symbol text* is described in the image’s header (items 1 and 2) and highlighted in **orange** color.
- 4) Later, in the internal sector a “control” must be performed, but the students use an “operation” symbol. Then, there is a *mistake in symbol class* which is highlighted in **orange** color and described as item 4 of the image’s header.


```
> Resultados de la evaluación del diagrama <curso_alumno_15.png>:
Nota: MAL
Observaciones:
- (1): Se indica "Documento" de tipo "OP0" pero debe ser de tipo "OC0".
- (2): Se indica "Documento" de tipo "OP0" pero debe ser de tipo "OC0".
- (3): Falta incluir "Archivo Definitivo".
- (4): Se indica "Operación" pero debe ser "Control".
- (5): Falta incluir "Flecha de Consulta Lógica".
- (6): Se indica "Documento de tipo "F0"" en una ubicación incorrecta.
- (7): Falta incluir "Decisión".
- (8): Falta incluir "documento" de tipo "F0".
+ (9): No corresponde incluir "Archivo Definitivo".
+ (10): No corresponde incluir "Control".
- (11): Falta incluir "Operación".
- (12): Falta incluir "Documento" de tipo "CH0".
+ (13): No corresponde incluir "Flecha de Consulta Lógica".
- (14): Se indica "Conector de Comienzo de Corte" pero debe ser "Archivo Definitivo".
- (15): Falta incluir "Conector de Comienzo de Corte".
```

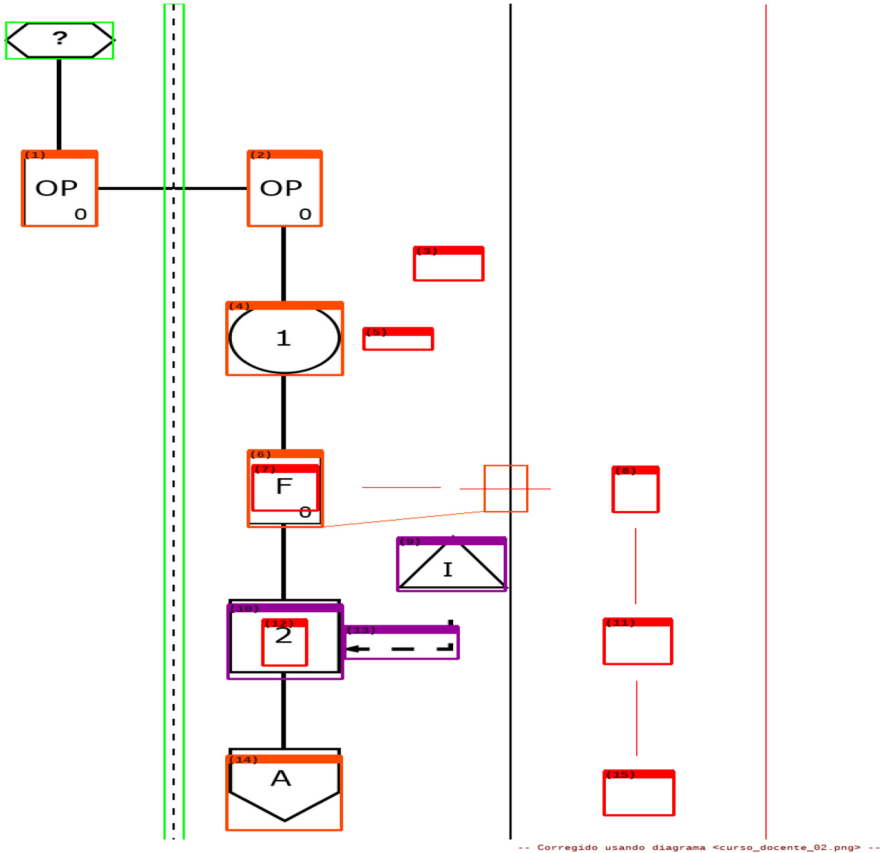
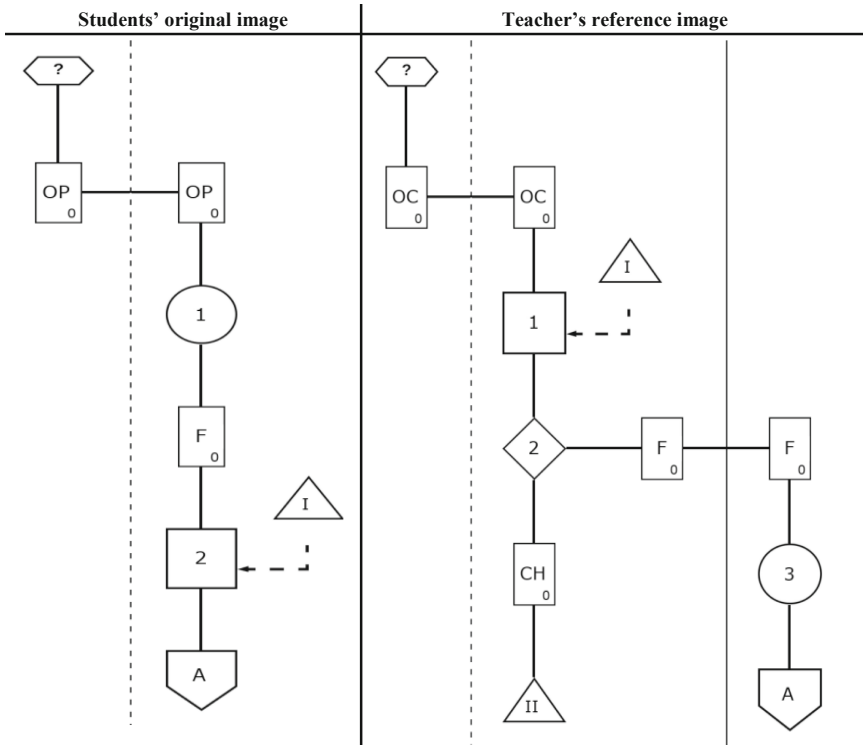


Fig. 2. Example of evaluating a students' diagram with multiple colors. (Color figure online)

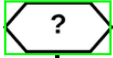
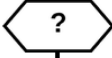




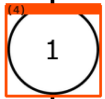
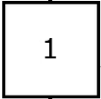

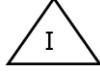


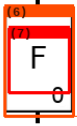
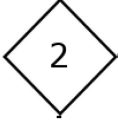
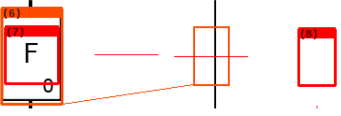
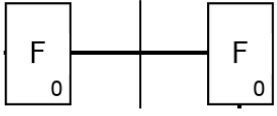




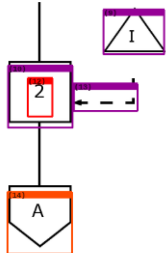
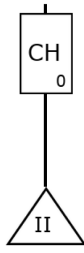
- 5) Moreover, the previously "control" must apply a "logical query" to a "permanent file", that the students have omitted. Then, these two symbols are marked as *missing* (items 3 and 5 of the image's header) and their corresponding positions in the students' image are highlighted in **red** color.
- 6) After a "control" there is always a "decision", but the students have not included it. The position of this "decision" should be where the document "F0" is located, but this document will be used afterwards. Therefore, the "decision" is also marked as

Table 3. Student and teacher diagrams used in the evaluation example.

missing (item 7 of the image's header) and its corresponding position is highlighted in **red** color.

- 7) The document "F0" is correctly indicated, but it is located in a wrong position of the diagram. Then, the document is marked as *mistake in symbol location* (item 6 of the image's header), highlighted in **orange** color and the correct position is also marked in the diagram with a thin **orange** box. On the other hand, this document also is transferred to another internal sector of the organization but this is *missing* in the students' image, so it is marked in item 8 and its corresponding position is highlighted in **red** color.
- 8) In the new internal sector of the organization, two more symbols are *missing* (an "operation" and a "cut start connector"). Therefore, both are marked as *missing* (items 11 and 15 of the image's header) and their corresponding positions are highlighted in **red** color.
- 9) Finally, in the old internal sector, there are three types of mistakes:
 - First, the document "CH0" is *missing*, so it is marked in item 12 of the image's header and its corresponding position is highlighted in **red** color.

Table 4. Analysis of the evaluation of the students' diagram example.

#	Students' Evaluation Result part	Corresponding Teacher's reference part
1		
2		
3		
4		
5		
		
6		
7		
8		
		
9		

- Secondly, the students have included a “cut start connector” instead of a “permanent file”, so this is marked as a *mistake in symbol class* (item 14) and highlighted in **orange** color.
- And third, there is a “control” that uses a “logical query” to a “permanent file” that are marked as *mistake of over indicated symbol* (items 9, 10 and 13) and highlighted in **violet** color.

This last mistake is interesting to be discussed in a little more detail. Although one might come to think that these three symbols correspond to the mistakes previously indicated in (3) and (4), it can be noted that their location are well below the diagram. If the Evaluation program will try to use them as a correct landmark, it would have to mark all the previous symbols as over indicated, and all the following symbols as missing. Since the idea is to try to reuse as much as possible the symbols available in the students’ diagram, it is considered that this kind of review is correctly performed.

4.3 Verification of the Robustness with “Noisy” Images

A question that has arisen during development is how the Evaluation program will behave with images that are not 100% “perfect” (i.e. images that present some degree of pixel error or noise). The main concern was directed to confirm if the previously trained Symbol Detection Model would be able to detect the symbols in this type of images. In order to perform this verification (as has been described in Sect. 4.1) a “noisy” version of the students’ images have been generated, and then evaluated by the software as a normal exercise diagram.

As a result, for each exercise there is a normal version and a “noisy” version that have been compared manually to determine if the Evaluation program generates different results. After reviewing all the pairs of images, differences have been detected in the evaluation of noisy images, but, surprisingly the differences were not due to the Symbol Detection Model.

Despite presenting different noise levels, the trained model is able to accurately detect the symbols present in the evaluated images. But, there is an issue with the Tesseract OCR Engine used to recognize the characters included in some symbols. For example, as it can be seen in the left column of Table 5, when the students’ diagram ‘04’ is evaluated using the teacher’s diagram ‘04’ as reference, obviously there are not mistakes (all the symbols are in **green** color) because they are the same image. However, when the “noisy” version of the student’s diagram is processed (right column of Table 5), three **orange** color mistakes are indicated in the “document” symbols. These errors are related to the *mistake in symbol text* because the OCR Engine cannot recognize correctly the letters and numbers in each symbol. Despite of the black pixels, a human can read that the documents’ strings are “RE-0”, “NP-0” and “OP-0”, but the OCR Engine recognizes them as “RE” (without any number), “NP.N8” and “OP.OO”. As the Evaluation program trusts in the strings provided by the OCR process, then mistakes are marked incorrectly.

Given that the evaluation of images with this degree of noise should not be normal in real diagrams, and that the OCR engine is an external element used by the software, it is considered that this issue does not invalidate the development or the tests presented

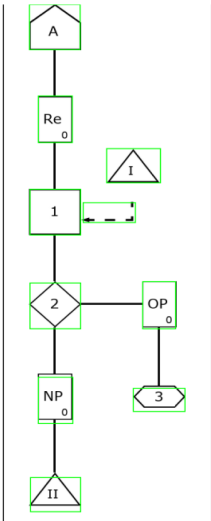
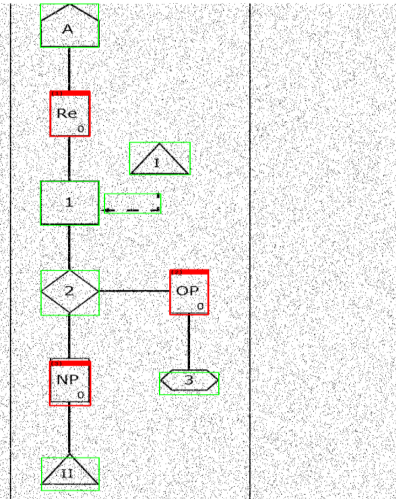
in this article. However, it has been decided to initiate a review of other available OCR engines that are more robust to be applied in a future version of the Evaluation program.

4.4 Discussions of the Evaluation Results

In this section, the main comments received by the expert teachers of the subject after reviewing the evaluations made by the software system are presented.

From their point of view, the software works in an acceptable manner, being able to correctly detect 100% of the symbols present in each image, which allows it to highlight different types of mistakes consistent with the reference image. Also, the observations indicated by the software system are considered as useful for the students. In fact, this will allow students to better understand their mistakes and try to avoid them in future exercises.

Table 5. Examples of comparing the evaluation of students' normal and "noisy" images.

Evaluation of students' image	Evaluation of students' "noisy" image
<p>> Resultados de la evaluación del diagrama <curso_alumno_04.png> Nota: EXCELENTE No hay observaciones.</p> 	<p>> Resultados de la evaluación del diagrama <curso_alumno_04_00a.png> Nota: BUENA Observaciones: - [1]: Se indica "Documento" de tipo "RE" pero debe ser de tipo "RE0". - [2]: Se indica "Documento" de tipo "OP_00" pero debe ser de tipo "OP0". - [3]: Se indica "Documento" de tipo "NP_00" pero debe ser de tipo "NP0".</p> 

On the other hand, it is believed that the grades assigned by the system in most of the cases are correct. From the expert teachers' point of view, it is considered that in around 80% of the cases, the assigned grade is consistent with the errors found. However, there are some errors that perhaps should be different, since there are mistakes that the software considers to be minor when in fact they should be serious mistakes. Such is the case of the "timing lines" that are important in a diagram since they indicate that the routine being plotted is performed every certain quantity if time or by certain specific condition (e.g. when a shortage of merchandise is detected). Given its importance, this should

be considered a serious mistake that indicates that the student does not understand that this routine is not always performed. Likewise, a “decision” should always be placed under a “control”. If this “decision” is not placed by the student, then it is a serious conceptual error and should be considered with more relevance than just considering that the symbol is missing. The same happens if the diagram is not finished with the corresponding end symbols that could be a “file”, a “destruction” action, a “process” or a “connector”. Therefore, it should be possible to differentiate them from other types of missing symbols.

Finally, it has been detected an issue associated to the location of some missing symbols. Although the location is correctly identified by the detection model, on some occasions, the location of the missing symbols is not 100% correct and might overlap with other symbol. This has been requested to be solved in future versions, since otherwise the students will not be able to understand the mistake they made.

5 Conclusions

In this paper we have described the tasks carried out for the implementation of a software system that allows the automatic evaluation of *Cursograma* diagrams made as part of the exercise of the students of an undergraduate course. So, although the domain of the problem belongs to the field of education, this work has more to do with the engineering and technological aspect of the application of Artificial Intelligence to solve complex problems.

First, a process of understanding the problem to be solved has been carried out, identifying the particularities of its domain and defining the requirements to be met. Taking all this into account, different technologies are studied that allow achieving the desired objective. As a result the use of object detection models based on Deep Neural Networks is selected. Then, the construction and testing of each of the components that are part of the software system are performed. Two of these components are auxiliary but necessary to achieve the third. On the one hand, a random diagram generator is developed. On the other, a model capable of recognizing the symbols is trained and validated with the diagrams previously generated. And finally, a program that applies this detection model to review and correct the students’ exercises. It can be observed that, to try to achieve a complex objective with many restrictions, the problem has been worked on in a modular way, with separate components that apply different approaches but converge to the same goal.

Finally, the results of the diagram evaluator are presented and analyzed. From the results obtained, it can be concluded that the software system works in an acceptable manner by being able to recognize different types of symbols, identify differences and correct four types of mistakes. But, although much progress has been made in principle, there is still a lot of work to be done before this software can be made operational for use by students. Consequently, as future lines of work, the following stand out:

- review other OCR engines that may have greater robustness when performing character recognition in noisy images;
- correct the issue associated with the location of missing symbols improving the system’s ability to avoid overlaps with other objects;

- identify and implement the different rules that used by teachers to assign the grades to exercises taking into account the seriousness of the mistake detected; and
- continue working on improving the robustness of the object Detection Model so that it supports a greater variety in the characteristics of the diagrams to be processed.

References

1. Pollo-Cattaneo, M.F.: *The Organization and its Information Systems*, 1st edn. Editorial CEIT, Buenos Aires (2017). ISBN 978-987-1978-36-6
2. IRAM (1974). IRAM 34503: Administrative procedures. General guidelines for the design of forms for graphic representation, 1st edn. Argentine Institute for Standardization and Certification (in force since 03/05/1974)
3. UTN-FRBA: Analytical program of the chair ‘Systems and Organizations’ - Plan 2008 (2008). <https://tinyurl.com/y7xx33y5>. Accessed May 2021
4. UTN: Curricular design of the Information Systems Engineering career - Plan 2008. Ordinance UTN-1150 (2008). <https://tinyurl.com/yb22zm2q>. Accessed June 2021
5. Cohen, P.R., Feigenbaum, E.A.: *The Handbook of Artificial Intelligence*, vol. 3. Butterworth-Heinemann (2014)
6. Vegega, C., Pytel, P., Pollo-Cattaneo, M.F.: Elicitation of requirements for the construction of predictive models based on intelligent systems in education. In: *Proceedings of XXV CACIC 2019*, pp. 980–989 (2019). ISBN 978-987-688-377-1
7. Chair SyO: Template of Symbols for use in Cursogramas to be used in the subject ‘Systems and Organizations’ of UTN FRBA (2020). <https://tinyurl.com/y6jdnfy3>. Accessed June 2021
8. Outair, A., Lyhyaoui, A., Tanana, M.: Towards an automatic evaluation of UML class diagrams by graph transformation. *Int. J. Comput. Appl.* **95**(21), 36–41 (2014). <https://doi.org/10.5120/16721-7063>
9. Lino, A. D.P., Rocha, A.: Automatic evaluation of ERD in e-learning environments. In: *2018 13th Iberian Conference on Information Systems and Technologies (CISTI)*, pp. 1–5 (2018)
10. Schäfer, B., Keuper, M., Stuckenschmidt, H.: Arrow R-CNN for handwritten diagram recognition. *Int. J. Doc. Anal. Recogn. (IJ DAR)* **24**(1–2), 3–17 (2021). <https://doi.org/10.1007/s10032-020-00361-1>
11. Anwer, S., El-Attar, M.: An evaluation of the statechart diagrams visual syntax. In: *2014 International Conference on Information Science & Applications (ICISA)*, pp. 1–4 (2014)
12. Vachharajani, V., Pareek, J., Gulabani, S.: Effective label matching for automatic evaluation of use-case diagrams. In: *2012 IEEE 4th International Conference on Technology for Education*, pp. 172–175 (July 2012)
13. Elyan, E., Jamieson, L., Ali-Gombe, A.: Deep learning for symbols detection and classification in engineering drawings. *Neural Netw.* **129**, 91–102 (2020)
14. Altun, O., Nooruldeen, O.: SKETRACK: stroke-based recognition of online hand-drawn sketches of arrow-connected diagrams and digital logic circuit diagrams. *Sci. Program.* **2019**, 1–17 (2019). <https://doi.org/10.1155/2019/6501264>
15. Alpaydin, E.: *Introduction to Machine Learning*. MIT Press (2014)
16. Schmidhuber, J.: Deep learning in neural networks: an overview. *Neural Netw.* **61**, 85–117 (2015)
17. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. *Nature* **521**(7553), 436 (2015)
18. Albawi, S., Mohammed, T.A., Al-Zawi, S.: Understanding of a convolutional neural network. In: *2017 International Conference on Engineering and Technology (ICET)*, pp. 1–6. IEEE (2017)

19. Zhou, D.X.: Universality of deep convolutional neural networks. *Appl. Comput. Harmon. Anal.* **48**(2), 787–794 (2020)
20. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) *ECCV 2014*. LNCS, vol. 8689, pp. 818–833. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10590-1_53
21. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J.: Rethinking the inception architecture for computer vision. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818–2826 (2016)
22. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: MobileNetV2: inverted residuals and linear bottlenecks. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520 (2018)
23. Huang, J., Rathod, V., et al.: Speed/accuracy trade-offs for modern convolutional object detectors. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7310–7311 (2017)
24. Khandelwal, R.: COCO and Pascal VOC data format for Object detection. Medium (2019). <https://tinyurl.com/y5fgyb9n>. Accessed June 2021
25. Smith, R.W.: History of the Tesseract OCR engine: what worked and what didn't. In: *Document Recognition and Retrieval XX*, vol. 8658, p. 865802. International Society for Optics and Photonics (February 2013)
26. GEMIS: Randomly Generated Cases with its Results (2021). <https://tinyurl.com/7k8mhrtw>. Accessed June 2021
27. Bovik, A.C. (ed.): *The Essential Guide to Image Processing*. Academic Press (2009)
28. Velosa, F., Florez, H.: Edge solution with machine learning and open data to interpret signs for people with visual disability. *CEUR Workshop Proc.* **2714**, 15–26 (2020)
29. Zuluaga, J.Y., Yepes-Calderon, F.: Tensor domain averaging in diffusion imaging of small animals to generate reliable tractography. *ParadigmPlus* **2**(1), 1–19 (2021)
30. Espinosa, C., Garcia, M., Fernando Yepes-Calderon, J., McComb, G., Florez, H.: Prostate cancer diagnosis automation using supervised artificial intelligence. a systematic literature review. In: Florez, H., Misra, S. (eds.) *ICAI 2020*. CCIS, vol. 1277, pp. 104–115. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-61702-8_8