Recognition of Handwritten Flowcharts using Convolutional Neural Networks

C. David Betancourt Montellano
Department of Computer Science,
National Polytechnic Institute,
UPIIZ,
P.C. 98160, Zacatecas,
Zacatecas, Mexico

C. Onder Francisco Campos Garcia
Department of Computer Science,
National Polytechnic Institute,
UPIIZ,
P.C. 98160, Zacatecas,
Zacatecas, Mexico

Roberto Oswaldo Cruz Leija
Department of Computer Science,
National Polytechnic Institute,
UPIIZ,
P.C. 98160, Zacatecas,
Zacatecas, Mexico

ABSTRACT
Currently, artificial vision is used in an endless number of tasks from domestic tasks to industrial and educational ones since with it those tasks can be streamlined because of having an automated process. This project explores the problem of handwritten information recovery, specifically the flowcharts used in the programming and designing of algorithms,approaching a solution with artificial vision, and proposes a pipeline able to recognize the elements of a handwritten flowchart using convolutional neural networks in order to generate code source in the C programming language equivalent to the recognized diagram, in addition the digitalized version of the flow diagram, thus automating the various tasks, having as a final result a file with *.c* extension with the source code, the compilation output and an image in *.png* format with the digitization of the diagram.

General Terms
Computer Vision, Compilers, Pattern Recognition.

Keywords
Convolutional neural network, flowchart, grammar analysis, image processing, object detection, sketches recognition

1. INTRODUCTION
A recognition pipeline was designed and implemented highlighting the use of convolutional neural networks (CNN) to test the accuracy in the recognition of figures, connectors, and text, being these the components of a flowchart [1] and with the help of grammar analysis verifies the structure of the flowchartto detecterrors on it. Seeking to see the result of the implemented pipeline usingPython (programming language), it receives as input a digital image of the hand drawn flowchart, processes the image, recognizes the elements of the flowchart, analyzes the grammar, and finally returns the result as an image of the flowchart once it has been reconstructed into a digital image andgenerates its equivalent C source code. This is only if the flowchart has been globally recognized with a correct structure. It is also important to say that the C generated programs follow the structured programming paradigm and that a defined set of symbols is used for the construction of the flowchart.

The specific reason for the development of the project is to explore the application of CNNs in the task of recognition of the elements of a handwritten flowchart under the *off-line* approach [2], and why is it thought that it is necessary to investigate? To seek higher accuracy in the overall recognition of the flowchart, and why is higher accuracy necessary? Because by ensuring a good enough effectiveness it would be possible to apply the solution on a software product that helps in the educational area, as in [3], or scientific areas. For example, it is known that nowadays most engineering curricula require students to be able to program at some level [4]. As a visual representation of data flow, flowcharts are useful for designing an algorithm and explaining it to others, and/or enabling effective collaboration. Thus, a flowchart can be used to spell out the logic behind a program before coding begins, as well as to organize the overall thinking and provide guidance when translating it into a programming language. In that way, students will focus more on problem-solving rather than on the syntax of a given programming language, reducing the learning curve in the early stages of learning programming.Currently there are similar and different approaches, but mostly for *online* approach [5, 6, 7] that wasalso a motivation for the realization of the project where the focus is the *offline* approach.

2. PIPELINE
The implemented pipeline for recognition of handwritten flowcharts (see Figure 1) receives an image as the input, then the image is processed by two detectors, the shape-connector detector, and the text detector. For text flow, the image is binarized and the text is located usingKeras OCR and it is classified with an implemented CNN+LSTM model, to enhance the precision of the text detector, the technique called continual learning is used,so after some trainings with the text style of a certain user the model will improve the recognition of text. On the other hand, for the flow of shapes and connectors unsharp masking is used to highlight image features then a Faster R-CNN model is used for object detection,the feature extraction is with backbone VGG-16 [8]. After the detection of all elements of the flowchart the bounding boxes (text, shape, and connector ones) are encapsulated in nodes anda directed graph is constructed using those nodes, the construction is supported using grammar analysis. At this point the flow of the pipeline can fail, maybe because the detected diagram has an incorrect structure or a bad or missing detection by the models. Finally, the outputs are the generated source code in C programming language, its compilation output, and the digital reconstructed diagram.

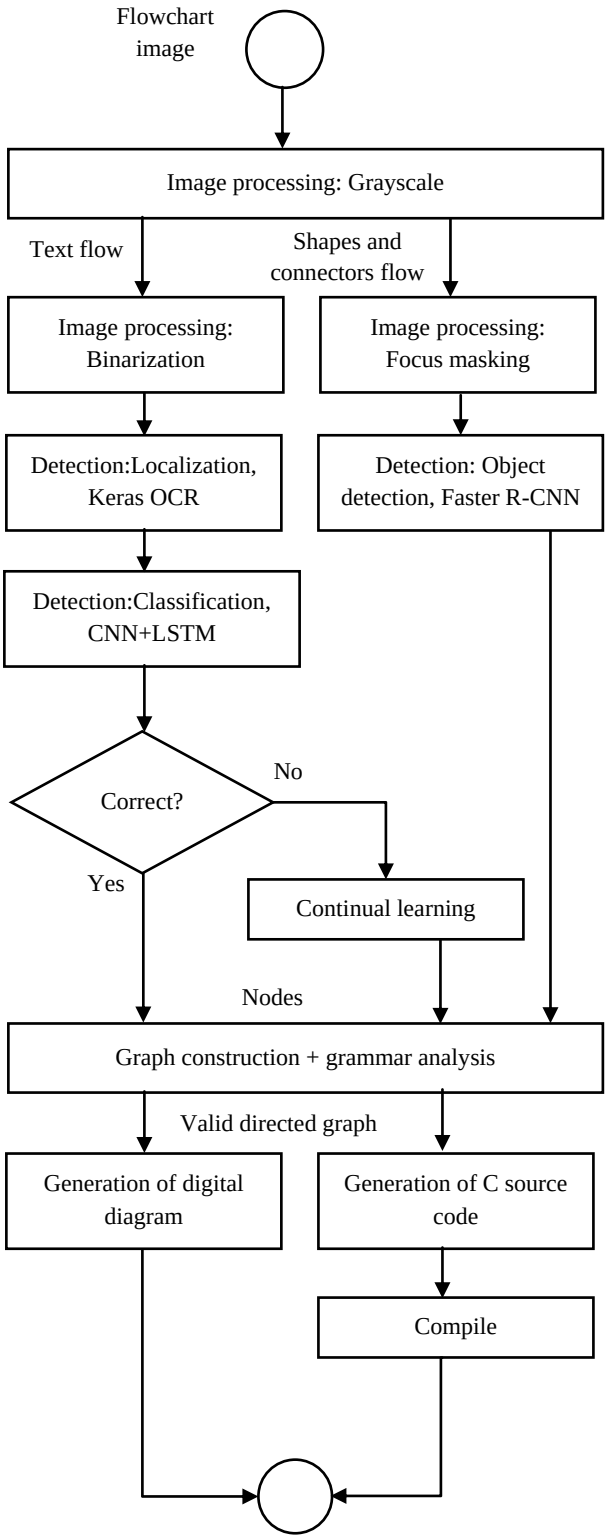

Fig1.Pipeline diagram. Source: Own elaboration

3. SHAPE MODEL

The object detection model used is Faster R-CNNwhich efficiently uses the convolutional feature layers, this model has a module called RPN with a classifier (object or non-object) and a regressor to adjust the coordinates of the bounding boxesThe backbone to extract features of the imageisVGG-16[9].

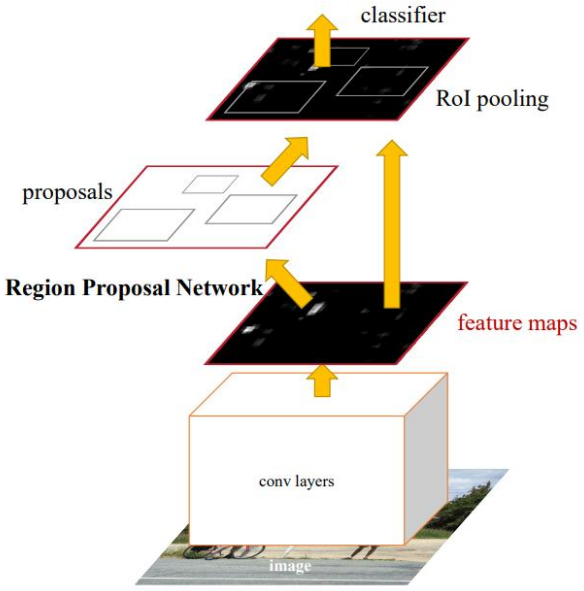

Fig 2.Faster R-CNN modelarchitecture. Source: [8].

3.1 DATASET *FLOWCHART 3B*

A dataset, called "Flowchart 3B", was compiled to develop the project. It was necessary to take many photos of shapes and connectors, moreover, draw several flowcharts as the representation of some known algorithms such as compute the factorial of a given natural number or calculate the nth element of the Fibonacci sequence. Altogether there are 775 images and their respective annotation files (in Extensible Markup Language format, XML), the Table 1 shows the number of items per class, and the Table2 shows the set of shapes and connectors allowed to compose the flowcharts, so there are 9 different classes to locate and classify in detection process by deep learning models. The dataset is oriented for the *off-line* approach and was created mainly with help of students of computer systems engineering bachelor's in 2019 and 2020 years. It is important to say that the backgrounds where the flowcharts were drawn are three different types: white-blank, grid and lined paper. The dataset is divided into two parts: Training and Validationsets.The dataset is available to download on Kagglehttps://www.kaggle.com/davbetm/flowchart-3b

Table 1. Dataset content, number of items per class. Source: Own elaboration.

Class name	Training set	Test set
start_end	838	208
scan	741	196
decision	761	187
print	798	204
process	882	206
arrow_line_up	760	185
arrow_line_down	1020	249
arrow_line_right	799	197
arrow_line_left	763	187
Total	**7362**	**1819**

Table 2.Flowchart elements. Source: Ownelaboration.

Class	Graphic representation
Startend	inicio fin
Scan (input data)	
Decision	
Print	
Process	
Arrowlineup	
Arrowlinedown	
Arrowlineright	
Arrowlineleft	

4. TEXT MODEL

The model used for text recognition is an implementation of the publishedpuigcerver *model*, see Figure 3.

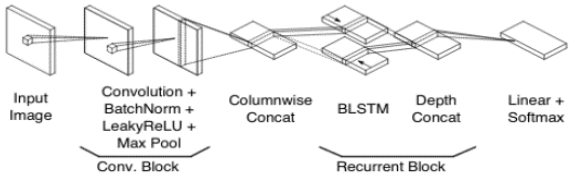

Fig 3.Architecture of the puigcerver *model* neural network. Source: [10]

The model architecture is constituted by convolution blocks where each block contains a two-dimensional convolutional layer, to reduce overfitting dropout is applied at the input of some convolutional layers after each convolutional layer batch normalization is applied, in addition to linear leaky rectifier units as activation function in the convolutional blocks. At the end, a maximum pooling layer to reduce the dimensionality of the input images. After the convolution blocks, recurrent blocks are applied which are formed by bidirectional 1D-LSTM layers, dropout is applied again at the output of the BLSTM blocks. Finally, after therecurrent blocks, each column must be assigned to an output label.

5. GRAPH BUILDING AND CODE GENERATION

The process to build the adjacency list of the graph that represents the given flowchart uses the following Conway diagram (see Figure 4) that describes the sequence that must follow the flow of the process given any node, considering the rules of the flowcharts.

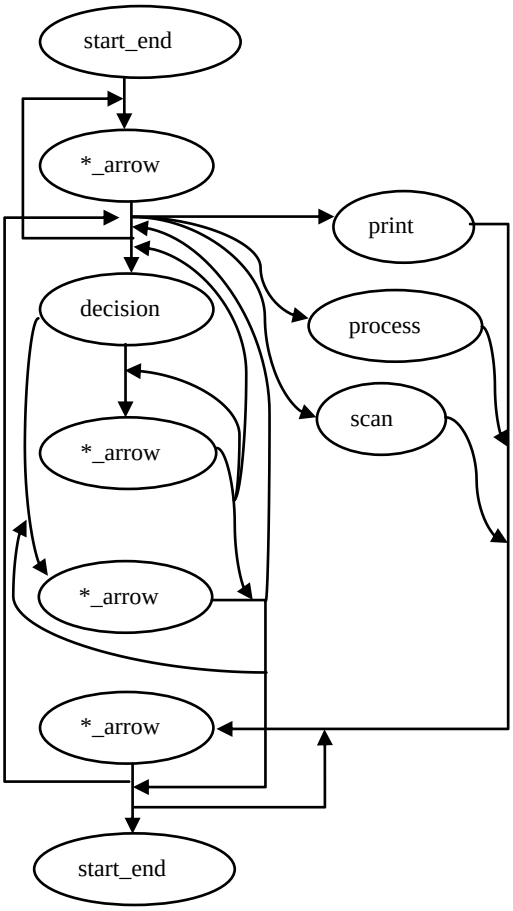

Fig 4.Conway diagram showing the grammar to construct the flowchart. Source: Own elaboration.

Given the nodes generated by the detection models, which can be either shape or text nodes, both nodes contain the points of the bounding box in the 2D image space, and a value that differs depending on the node, if it is a shape node contains the label given by the shape model, on the other hand if it is a text node then contains the text recognized by the text model, the first step is to find and append to shape nodes the text value of all the text nodes whose points are inside the shape nodes or if the shape node are labeled by "arrow" the text node may be in the vicinity of the shape node, this process generate a new set of nodes whose data contains the bounding box position, the label and now if it is the case also contain a text value, that allows to generate the graph using only a set of nodes, the next step is find the node labeled by "start_end" and text value with “inicio” or “start” (Spanish or English language), once found, an iterative process is started, for every node there are possible successors which are selected following the intrinsic rules from handwritten flowchart described in Conway Diagram, the process finishes when all the nodes are traversed.

6. RESULTS

In this section, we present the results of the implementation

and tuning of each element of the pipeline, the results of the deep learning models used in the project on their own, as well as the algorithm, used to build the graph and the source code in C. In this way, we can measure the performance of the elements on their own and toconsider the pros and cons.

6.1 SHAPE MODEL RESULTS

The following plot shows the average accuracy obtained by each classusing validation dataset, taking as true positive when the bounding box of the object recognized has an Intersection over Union (IoU) of 50% with the ground truth. The arrow classes of up, down, and left have the relative lowest values, all other classes have almost perfect average precision.

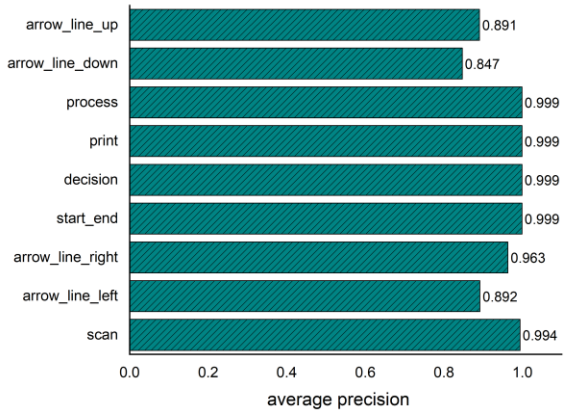

Fig 5.Average precision per class. Source: Own elaboration

Most (75%) of the tested images from a set of 56 flowcharts images were completely recognized. The model could not recognize mainly arrows and some flowcharts as a special case, such as colored lines and faint strokes, that is a signal that with a better data centric approach it could be better.

6.2 TEXT MODEL RESULTS

About classification of text, the dataset used was IAM [11], and evaluating the model the precision achieved was 66.7% and with Character error rate (CER) metricwas 91.8%. Detection of text was a hard task to solve, in order to improve the rate of completed executions of pipeline a technique called continual learning was used.

6.3 PIPELINE RESULTS

The performance of the pipeline is measured with compliance of all the parts in it, therefore the output must be a file with C source code, its compilation output, and an image with a digital version of the flowchart in the input image.

Of the total number of tests performed, 60% of the cases generated the desired directed graph, of which 75% of these generated the expected output.Figure 6shows an example of an input image, the outputs of the pipeline for that input are: the C code generated code (see Figure 8), the corresponding digital flowchart image (see Figure 7) and the compilation output that was correct so there are no output errors.

Table 3. Tabular results as a big picture of pipeline executions. Source: Own elaboration.

Result	% of tests	% of tests with complete execution
Correct graph	60	**45**
Incorrect graph	40	0

This brings a useful insight, the fact that even that a valid directed graph was generated, other errors raised by different causes such as run time errors generating source code, those lowered the percentage in 15%.

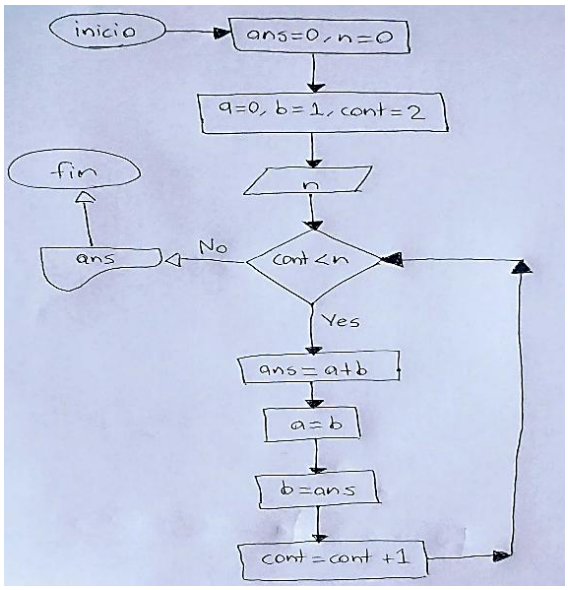

Fig 6. Input image, flowchart describing a Fibonacci sequence generator. Source: Own elaboration

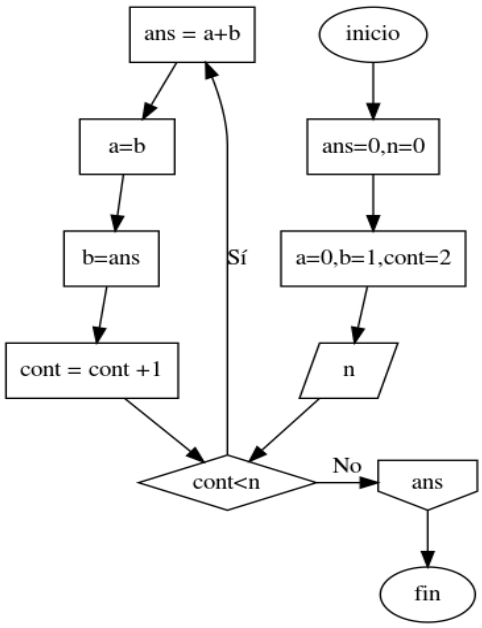

Fig 7. Output image, digital flowchart. Source: Own elaboration

There is space for presentation improvements, for instance, using the same type of arrows as in the input image and the spatial disposition of elements.Curved linesare used on the output image,however, the shape model was not trained to recognize curved or more complex/ freehand oneslike in [12].

```
#include<stdio.h>

intmain(){
    intans=0, n=0;
     int a=0, b=1, count=2;
    scanf("%d", &n);
    while(count<n){
        ans=a+b;
        a=b;
        b=ans;
        count=count+1;
    }
    printf("%d", ans);
    return0;
}
```

Fig 8. Source code output.Source: Own elaboration

Looking at the code generated, it could be improved in code style, such as inserting blank spaces to separate control structures from the rest of the code or when declaring variables at both sides of equal operator.

7. CONCLUSIONS

In this paper it was proposed a pipeline for the task of handwriting flowchart recognition to obtain their respective C-code source and a digitized flowchart image.An object detection model (Faster R-CNN) was trained to recognize and classify the shapes on the flowchart, *Keras OCR*was used to localize text anda CNN+LSTM model was trained for text classification. Experimental results show that the designed and implemented pipeline performs its function andthat the implemented deep learning models using convolutional neural networkscan detect the components of a handwritten flowchart so it was demonstrated that is possible to construct software applications, because the whole implementation it can be seen as an image to code traductor, taking advantage of the graphs as mathematical structures.

In future work, it could be improved the generated digital flowchart, text detection and the data used, for the next reasons respectively,it was used a “print” shape different from the defined set and the spatial arrangement of the shapes could be better; the second one is, two different models were used, instead of a unified one, which make the implementation slow and costly when training; and finally, because the classification model was trained with images taken in controlled conditionscompared to those used in real life where there are also faint strokes and colored lines, those cases caused low precision because they are causing data drift.

8. ACKNOWLEDGMENTS

Thanks to all colleagues and friends who helped us a lot to create the dataset, to computer science professors in UPIIZ, especially to our advisors M. Sc. Roberto Oswaldo Cruz and M. Ed. Karina Rodríguez Mejía. Moreover, not less important, to our families by all the support while planning, designing, and implementing this challenging project to us. You can find the repository of code on GitHub: https://github.com/dbetm/handwritten-flowchart-with-cnn

9. REFERENCES

[1] O. CairóBattistutti, Metodología de la programación, 3rd ed. México, D.F.: Alfaomega, 1995, pp. 3, 4-8.

[2] W. Szwoch and M. Mucha, “Recognition of Hand Drawn Flowcharts” Advances in Intelligent Systems and Computing Image Processing and Communications Challenges 4, vol. 184, pp. 65–72, 2013.

[3] J. I. Herrera Camara, "Flow2Code: from hand-drawn flowcharts to code execution", master thesis, Texas A&M University, 2017.

[4] M. Mccracken, et al., “A multi-national, multi-institutional study of assessment of programming skills of first-year CS students,” Working group reports from ITiCSE on Innovation and technology in computer science education - ITiCSE-WGR 01, 2001.

[5] Wang, C., Mouchère, H., Viard-Gaudin, C., Jin, L.: Combined segmentation and recognition of online handwritten diagrams with high order Markov random field, 2016.

[6] Wu, J., Wang, C., Zhang, L., Rui, Y.: Offline sketch parsing via shapeness estimation.2015.

[7] Yun, X.L., Zhang, Y.M., Ye, J.Y., Liu, C.L.: Online handwritten diagram recognition with graph attention networks. 2019.

[8] Faster R-CNN: Towards real-time object detection with region proposal networks, S Ren, K He, R Girshick, J Sun - Advances in neural information processing systems, 2015.

[9] Karen Simonyan, Andrew Zisserman,Very DeepConvolutional Networks for Large-Scale Image Recognition, 2015.

[10] J. Puigcerver, "Are Multidimensional Recurrent Layers Really Necessary for Handwritten Text Recognition?" 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), Kyoto, 2017, pp. 67-72, doi: 10.1109/ICDAR.2017.20.

[11] U. Marti and H. Bunke. The IAM-database: An English Sentence Database forOff-line Handwriting Recognition. Int. Journal on Document Analysis and Recognition,Volume 5, pages 39 - 46, 2002.

[12] Schäfer, B., Keuper, M. &Stuckenschmidt, H. Arrow R-CNN for handwritten diagram recognition. IJDAR 24, 3–17, 2021.