

Received 28 June 2022, accepted 13 July 2022, date of publication 19 July 2022, date of current version 25 July 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3192467

## RESEARCH ARTICLE

# Handwritten Logic Circuits Analysis Using the YOLO Network and a New Boundary Tracking Algorithm

SOMAIIEH AMRAEE<sup>1</sup>, MARYAM CHINIPARDAZ<sup>1</sup>, MOHAMMADALI CHAROOSAEI<sup>1</sup>,  
AND MOHAMMAD AMIN MIRZAEI

Department of Electrical and Computer Engineering, Jundi-Shapur University of Technology, Dezful 64615/334, Iran

Corresponding author: Somaieh Amraee (s.amraee@jsu.ac.ir)

**ABSTRACT** Handwriting analysis has been addressed by researchers for decades, and many advances were achieved in understanding handwritten texts so far. However, some applications have been rarely discussed. One of these applications that has received less attention is the understanding and analyzing of handwritten circuits. Today, with the widespread use of intelligent tools in engineering and educational processes, the need for new and accurate solutions for processing such handwritings is felt more than ever. This paper presents a new method to analyze handwritten logic circuits. In this method, circuit components are first identified using a deep neural network based on YOLO. Then, the connection among these components is recognized using a new simple boundary tracking method. Then, the binary function related to the handwritten circuit is obtained. Finally, the truth table of the logic circuit is generated. We have also created a set of various handwritten logic circuits called JSU-HWLC. The results of the experiments show the proper performance of the proposed method on the collected dataset. The experiments demonstrated that the YOLO algorithm achieved better results than other deep learning methods such as faster R-CNN, Detectron2, and RetinaNet. For this reason, YOLO has been used to identify logic gates in the proposed system.

**INDEX TERMS** Handwritten logic circuit, deep learning, YOLO, boundary tracking.

## I. INTRODUCTION

Handwritten documents are one of the most important ways to record information. The processing of handwritten texts has been of particular importance for decades. These processes are performed to authenticate, understand handwritten content, classify, etc. [1]– [9]. There are different types of manuscripts, one of which is circuits drawn by engineers, professors, or students. The circuit diagram consists of various symbols called circuit components that determine the function of that circuit. Recognizing circuit components and then understanding circuit performance by machine vision algorithms can be used in various areas, including faster solutions to engineering problems as well as the production of intelligent graphics boards. Drawing handwritten logic

circuits is frequently the initial phase in the design process in digital system design, which includes the symbolic depiction of gates and other logic components, as well as their connections. To date, circuit designers have had to physically enter all of the manual circuit data into the computer, and this method, in addition to being time-consuming, is likely to lead to errors. For this reason, in recent years, researchers were interested in providing new solutions to create a system, including the process of automatic segmentation and recognition of parts in handwritten documents to create computer-aided manufacturing (CAD/CAM) systems.

In a comparative study on recognition processes of different symbols, including widely used symbols in engineering documents, different methods of symbol recognition are divided into four general categories [1]. This study has investigated statistical, structural, syntactic, and hybrid methods, but it does not mention new methods based on

The associate editor coordinating the review of this manuscript and approving it for publication was Yizhang Jiang<sup>1</sup>.

deep learning networks. Another work [2] has provided a method for identifying components of manual electrical and electronic circuits, including analog and digital components. In this method, after performing the desired preprocessing, a set of texture-describing features and shape-based features including the histogram of oriented gradients (HOG), centroid distance, tangent angle, and chain code histogram, are extracted. These features are then optimized using a feature selection algorithm called Relief. The classification of circuit components is performed using sequential minimal optimization (SMO). This method only identifies circuit components and their classification and does not involve recognizing connections as well as understanding circuit performance.

In the method proposed in [3], a combination of the local binary pattern (LBP) and statistical features based on pixel depth is used to detect the circuit components. Then, the components are classified using a support vector machine (SVM) model. After identifying the components, the proposed method subsequently uses their position to determine the type of circuit. To do this, it displays a sequence of components known as a string. This string representation of the circuit is given to the finite state machine (FSM) to identify the type of circuit.

The segmentation process is performed in [4] using a series of morphological operations on binary images and differentiation among three categories of components (closed form, components with connected lines, discrete components). Each segmented component is described by calculating the HOG, while the classification is performed using SVM.

In recent years, deep neural networks have made significant progress and have yielded acceptable results in many applications, including object detection and recognition [10]–[16]. A system that accepts sketched logic circuits as input recognizes various components of the circuit using R-CNN and converts the sketch-based circuit into a 2-D formal graphical format is provided in [10]. This system produces a digitized version of hand-drawn sketched images, but it seems that this system has not yet reached maturity. Its data set consists of only AND, OR, and NOR gates, and assumes that the original circuits are carefully drawn, with sharp corners, and without discontinuities. A two-stage CNN-based electrical and electronic component detection system has been provided in [15]. In the first step, a CNN-based clustering algorithm is implemented to get all the components that are similar components in shape and structure into one cluster. Then, the circuit components belonging to each cluster are classified into their actual classes. This approach is likewise confined to circuit component identification and does not include connection analysis or circuit performance knowledge.

The YOLO [16] algorithm was first introduced in 2016 to detect objects with high speed and accuracy. This method introduced a new structure for object recognition systems and has received much attention accordingly. Hence, different versions of YOLO were implemented in various applications [17]–[25]. YOLO stands for “You Only Look Once”. The term refers to the ability of the human visual system

to detect objects at a glance. Therefore, the YOLO object recognition system is designed to provide a method like that of the human visual system. The first version of YOLO consists of a 24-layer convolutional neural network for feature extraction as well as two fully connected layers for predicting the probability and coordinates of objects.

The present paper aims to develop an intelligent system that can detect the types of digital gates, apart from understanding the operation of the handwritten logic circuit and produces its truth table. The circuit components detection step is performed using in-depth learning based on the YOLO algorithm. Then, a simple boundary tracking method is proposed to detect the connection wires in the circuit, and finally the binary function and the corresponding truth table are generated. Such a system will be helpful at both educational and industrial levels. Although this program was developed to analyze logic circuits, it can be generalized to use in many other fields and to process various types of engineering circuits. To test the efficiency of the proposed method, a diverse set of handwritten logic circuits called JSU-HWLC has been created. The experimental results showed that the proposed method is useful for turning the drawn circuits into suitable information for higher-level designs. The rest of this paper is structured as follows: the main stages of the proposed system are described in Section II. Section III covers the experimental results on the JSU-HWLC dataset, and Section IV is dedicated to conclusions and recommendations for future research.

## II. THE PROPOSED SYSTEM

The proposed system consists of several steps, which are described below. First, the required collection of photos is created. Different individuals have sketched logic circuits in the picture collection. These photos were tagged to prepare the YOLO network for training. The connecting lines among the various components of the circuit are discovered once the YOLO algorithm has identified them. By identifying the gates and determining the relevant connections wires, the binary function of the original circuit is generated. Finally, its truth table is drawn. The following is an explanation of each of these steps.

### STEP 1- PREPARING THE DATASET

The set of images produced in this article is called JSU-HWLC, which contains 240 handwritten logic circuits. These circuits were drawn by computer engineering students of the Jundi-Shapur University of Technology of Dezful. To improve the detection of the proposed system, these students were asked to denote the input and output signals with a special symbol. Figure 1 shows some examples of these images. In this figure, the input pins are represented by a tiny circle, while a little square represents the output pins. A total of 240 circuits are included in the collection of photos created, with 120, 50, and 70 images utilized for training, validation, and testing, respectively. After collecting the total dataset, some initial preprocessing, such as resizing to

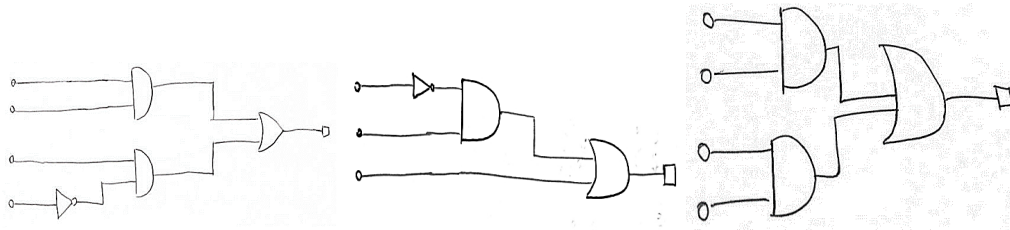


FIGURE 1. Some examples of the handwritten circuits in the JSU-HWLC.

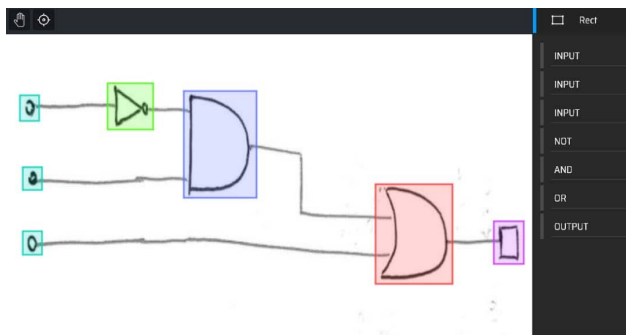


FIGURE 2. Overview of MAKESENSE script.

640 × 640 pixels along with converting to gray-scale images, is performed on them. Table 1 describes the number of images and the number of components in the JSU-HWLC dataset.<sup>1</sup>

### STEP 2- IMAGE ANNOTATION

In this step, we should label all the images that we have prepared for the training and validation processes, and introduce all the gates to the program with labels. Drawn circles and squares that symbolize the inputs and outputs must also be labeled and identified as individual objects. At this stage, the available scripts can be used for labeling. These scripts can generate appropriate input information for the YOLO algorithm. For this purpose, a free and simple<sup>2</sup> script [1] was used in our proposed system for this purpose. In MAKESENSE script, after entering various labels such as AND, OR, NOT, XOR, INPUT, and OUTPUT, the corresponding label can be selected by drawing a rectangle around each circuit component. This process is repeated for all components of each circuit and all training images. Figure 2 shows an overview of this program. The program creates a text file for each image, which provides the coordinate of the circuit components as well as their labels for the YOLO algorithm.

### STEP 3- YOLO ALGORITHM

After preparing the input data, it is time to train the YOLO neural network. The proposed system is implemented on Google Colab using the YOLOv5s<sup>3</sup> version coded in Python. YOLO training is done using the JSU-HWLC dataset. For

<sup>1</sup>JSU-HWLC is available on <http://dl2.jsu.ac.ir/index.php/s/NjUkaaE7JHsIGB6>

<sup>2</sup><https://www.makesense.ai>

<sup>3</sup><https://colab.research.google.com/github/ultralytics/yolov5/blob/master/tutorial.ipynb>

this purpose, it is necessary to upload the desired image set on Google Colab and then train the deep network of YOLO. The output structure of the trained network can be used to run on a test set. The gates were recognized with great precision, and were represented by boxes with various colors and labels. In the experiments, the accuracy of YOLO for digital component categorization was substantially higher than traditional approaches like support vector machine (SVM) or K-nearest neighbor. (KNN). The results of this comparison are presented in Section III. Also, as will be described in Section III, the YOLO algorithm achieved better results than other deep learning methods such as Faster R-CNN [12], Detectron2 [14], [26], and RetinaNet [13].

### STEP 4- CIRCUIT COMPONENTS RECOGNITION

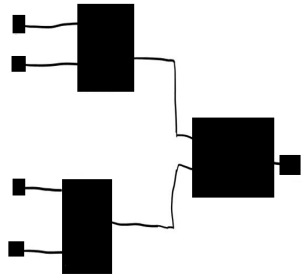
One of the YOLO outputs is a text file that lists the coordinates of the identified components. In this step, we put a black rectangle in the original image instead of each component using this file. Also, at this step it is possible to manually correct the misclassified gates of the previous step. This is done by modifying the class number in the output text file of YOLO. Putting a black rectangle instead of each component simplifies the initial image and prepares it for the next step. It should be noted that for each black rectangle, its corresponding class is recorded in the YOLO output file, and we can extract the type of components from this file if needed. Figure 3-A shows an example of the resulting image of YOLO after placing black rectangles. Figure 3-B shows the YOLO text file. Each row in this file represents one of the components identified by YOLO. The first column indicates the component's type, while the remaining columns describe the rectangle's center point, length and breadth, and probability of object identification, respectively.

### STEP 5- CONNECTION LINES DETECTION

In this step, first, the output of the previous step (Figure 3-B) is converted into a binary image; then, the quality of the binary image is improved using the edge detection algorithm and morphology operators to eliminate unwanted noise as much as possible. Also, morphological operators are applied to binary images to correct circuit breaks and missing connections. According to the experiments, four dilation operators were used on binary images, filling the lines and smoothing the image. Figure 4 shows an example of a prepared binary image.

TABLE 1. The JSU-HWLC image set.

	Number of images	Number of components						
		IN	AND	OR	NOT	XOR	OUT	Total
Train set	120	535	158	140	138	91	120	1182
Test set	70	300	94	72	95	47	70	678
Validation set	50	215	73	45	54	39	50	476
Total	240	1050	325	257	287	177	240	2336



```

0 0.0742188 0.649219 0.0640625 0.0578125 0.723832
0 0.0703125 0.825 0.065625 0.0625 0.777754
0 0.0703125 0.169531 0.075 0.0734375 0.810944
0 0.0664062 0.276563 0.0703125 0.06875 0.811486
5 0.896094 0.594531 0.0671875 0.0828125 0.850249
2 0.667188 0.558594 0.14375 0.201562 0.914881
1 0.284375 0.75625 0.165625 0.2875 0.927558
1 0.341406 0.246875 0.176563 0.26875 0.93324
    
```

(a) Drawing a black rectangle instead of the gates.

(b) YOLO output text.

FIGURE 3. Identifying circuit components.

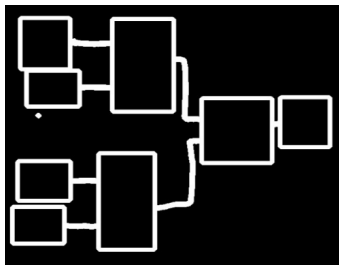


FIGURE 4. Binary image after morphology operation.

After preparing the binary image, it is time to recognize the connection wires. By performing morphological operations, the thickness of the lines has become more than one pixel. To determine how the gates are connected, we do the following for each gate:

- 1) As shown in the yellow circle in Figure 5 and its magnification on the left side of the image, we start from the bottom left corner of each component. Hence, the green pixel in Figure 5, indicated by point 1, will be the starting point of the tracking. The coordinates of this point are obtained from the YOLO output file.
- 2) Now assuming the circuit is drawn from left to right, we know that the inputs are connected to the left side of the rectangle. Starting from point 1, we move to the left on the blue path to reach the first black pixel on the border of black and white pixels (point 2).
- 3) From point 2, we change the direction upwards and continue the red path until we reach the first white pixel. This pixel is the entrance point of the first input signal (point 3). From now on, in this paper, these pixels are called input points.
- 4) If the label obtained by YOLO says that the gate has more than one input, the process continues to find the other input signals. Thus, we follow the path from

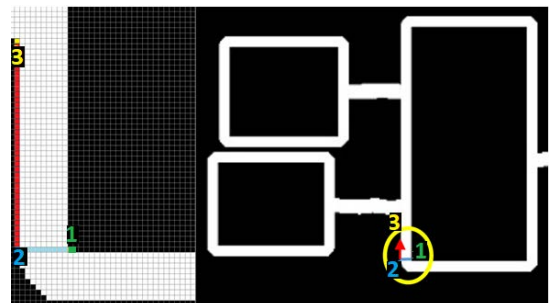


FIGURE 5. Detection of the first (lowest) input signal.

point 3 (in Figure 5) upwards and along the white pixels of the first input line, as shown in the blue path of Figure 6. Then, we continue upwards (red path) as before to reach the next input point.

We perform these four steps for all gates to obtain the input points specified in Figure 7-A. After finding the input points for all components of circuits, we start moving from the lower right corner of each component using the boundary tracking algorithm and move to the right to get out of the rectangle. As shown in Figure 7-B, we follow the wire boundary at black background pixels. In this way, we track the black pixels attached to the white line to reach an endpoint. A close view of this method is shown in Figure 7-C. As it can be seen in this figure, the starting point was shown by the green pixel at the lower right corner of a component, and the blue path represents the white pixels that must be passed to exit the rectangle. By hitting the first black pixel, the boundary between the white and black pixels is followed (red path) until it reaches the point that was specified as the input point of another component (yellow pixel).

After reaching the input point, the boundary tracking algorithm determines the link between the two circuit compo-

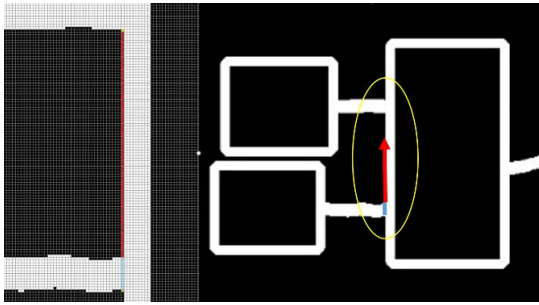


FIGURE 6. Detection of other input signals.

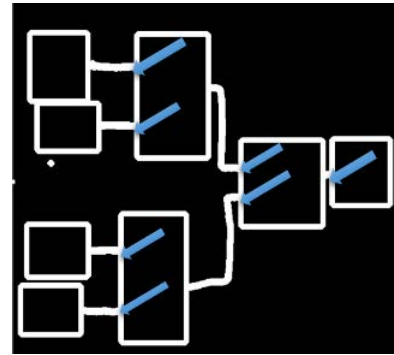
TABLE 2. The truth table of the logic circuit.

Inputs <i>xyzw</i>	Gate 6 $x \cdot y$	Gate 8 $z \cdot w$	Output $F(x, y, z, w)$
0000	0	0	0
0001	0	0	0
0010	0	0	0
0011	0	1	1
0100	0	0	0
0101	0	0	0
0110	0	0	0
0111	0	1	1
1000	0	0	0
1001	0	0	0
1010	0	0	0
1011	0	1	1
1100	1	0	1
1101	1	0	1
1110	1	0	1
1111	1	1	1

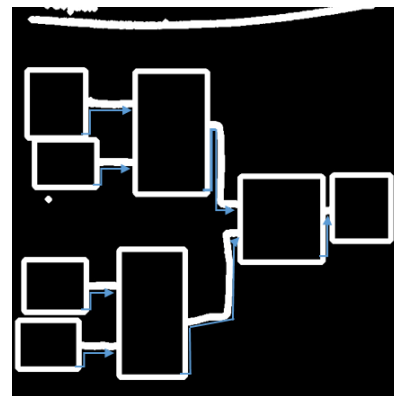
nents. The input of the second component, which has an input point, is linked to the output of the first component, from whence the navigation began. The relationship of all gates, input, and output signals can be obtained in the same way. For example, in Figure 7-D if we start from component 1 and use the boundary tracking algorithm, we reach an input point of component 2. So, we can say that the output of gate 1 is used as one of the input signals of gate 2, and because gate 1 shows one of the primary inputs of the circuit, input 1 is connected directly to gate 2. Repeating this operation for all circuit components leads to finding all circuit connections. Figure 8 shows the steps for extracting component input points (Figure 8-A) and finding interconnected components (Figure 8-B).

**STEP 6- BINARY FUNCTION**

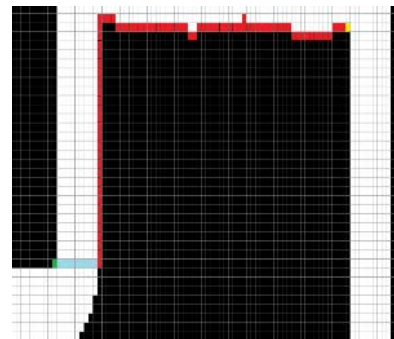
After finding the connection wires among the components of the circuit, the binary function of the circuit can be obtained by following the connections from the input pins of the circuit to the output pin. For this purpose, we start from the output components and find the gates connected to them based on the connections obtained from the previous step. We performed this procedure recursively for each gate. The boundary condition in this recursive algorithm is the circuit input pins. Thus, the corresponding function is obtained, which can be written in an infix form. In Figure 9-A, each component is shown by a number that is the row number of the corresponding text



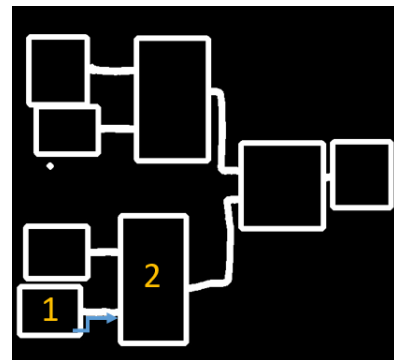
(a)



(b)



(c)



(d)

FIGURE 7. Connection wires detection. (a) Finding input points of each component. (b) Following the wire boundary to reach the endpoint. (c) Close view of following the wire boundary. (d) Determining the link and relationship between components.

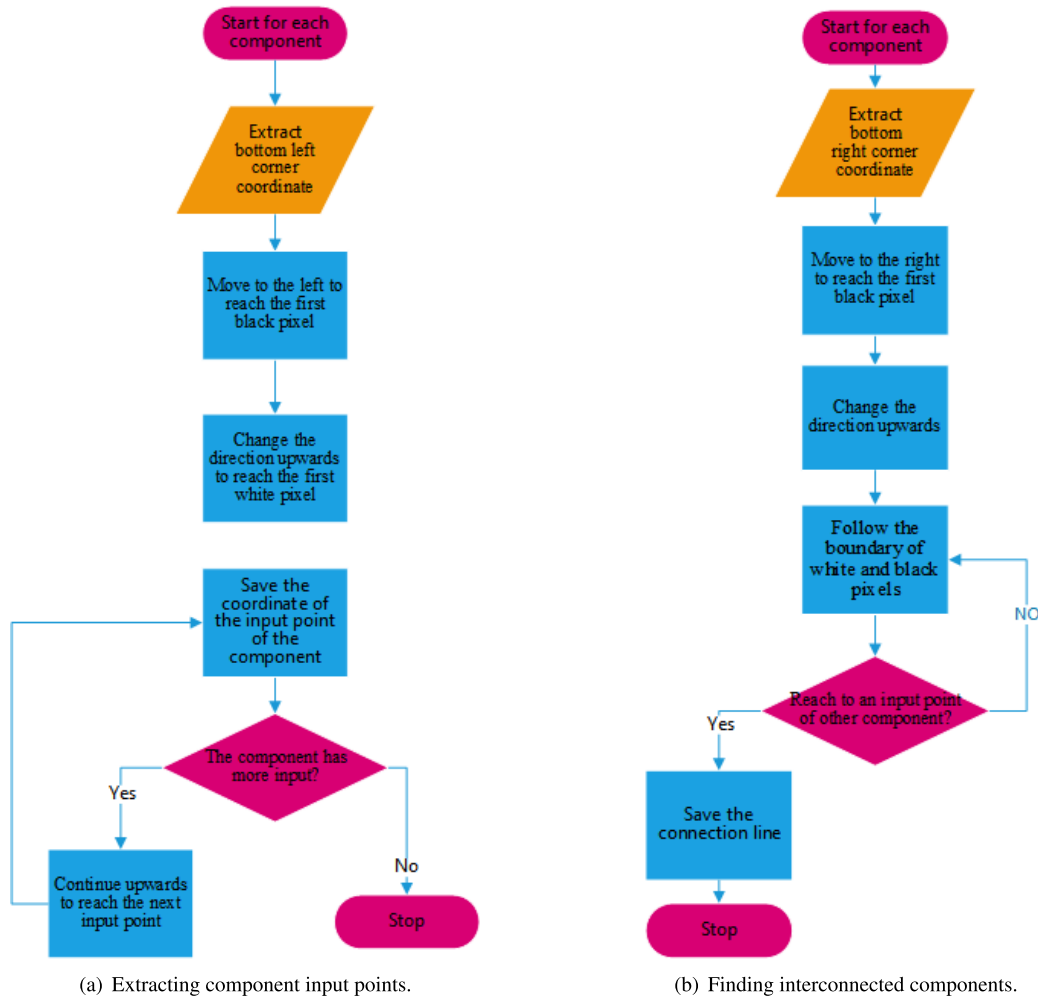


FIGURE 8. Flowchart of connection lines detection.

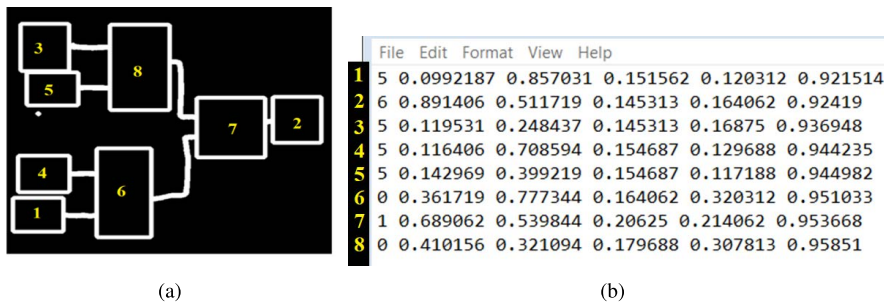


FIGURE 9. Numbering components to generate the binary function. (a) Visualizing each component in the circuit by its class number. (b) YOLO labeling output file.

file of YOLO in Figure 9-B. The original circuit is shown in the right image of Figure 1. The following connections are extracted from this example circuit:

$$2 \rightarrow 7 \quad 7 \rightarrow 8 \quad 7 \rightarrow 6 \quad 6 \rightarrow 4$$

$$6 \rightarrow 1 \quad 8 \rightarrow 3 \quad 8 \rightarrow 5$$

In Figure 9, for example, gate 6 is a component with the class number of 0 in the sixth row of the YOLO output file (Figure 9-B), which represents the AND gate. Now, we can say that because component number 2 in Figure 9-A is the output pin of the circuit, its value is the result of the binary

function, which itself is the result of performing OR on the outputs of gates 6 and 8. These gates are performing AND operation on input pins (see Figure 4); therefore, the binary function of this circuit can be written as the following infix expression:

$$f = ((1) \text{ AND } (4)) \text{ OR } ((3) \text{ AND } (5)) \quad (1)$$

By replacing the input pins with appropriate variables, and using symbols such as '+' and '.' instead of AND and OR, the function  $f$  is expressed as follows:

$$f(x, y, w, z) = (x \cdot y) + (w \cdot z) \quad (2)$$

**TABLE 3. Confusion matrix of YOLO in comparison with the other deep learning methods.**

Detectron2		INPUT	AND	OR	NOT	XOR	OUTPUT	ERROR
	INPUT (300)	300	0	0	0	0	0	11
	AND (94)	0	94	0	0	0	0	0
	OR(72)	0	0	71	0	0	0	0
	NOT(95)	0	0	0	92	0	0	0
	XOR(47)	0	0	0	0	47	0	1
	OUTPUT (70)	0	0	0	0	0	70	0
Faster R-CNN		INPUT	AND	OR	NOT	XOR	OUTPUT	ERROR
	INPUT(300)	300	0	0	0	0	0	1
	AND (94)	0	93	0	0	0	0	0
	OR(72)	0	0	70	0	0	0	0
	NOT(95)	0	0	0	92	0	0	0
	XOR (47)	0	0	0	0	45	0	0
	OUTPUT (70)	0	0	0	0	0	70	1
RetinaNet		INPUT	AND	OR	NOT	XOR	OUTPUT	ERROR
	INPUT(300)	291	1	0	0	0	0	2
	AND (94)	0	93	1	0	0	0	1
	OR (72)	0	0	70	0	1	0	0
	NOT (95)	0	0	3	92	0	0	2
	XOR (47)	0	0	2	0	45	0	0
	OUTPUT (70)	0	0	0	0	0	70	1
YOLO		INPUT	AND	OR	NOT	XOR	OUTPUT	ERROR
	INPUT(300)	300	0	0	0	0	0	0
	AND (94)	0	94	0	0	0	0	0
	OR (72)	0	0	72	0	0	0	0
	NOT (95)	0	0	0	95	0	0	0
	XOR (47)	0	0	1	0	46	0	0
	OUTPUT (70)	0	0	0	0	0	70	0

**TABLE 4. Confusion matrices of traditional feature-based classifiers.**

Classifier		INPUT	AND	OR	NOT	XOR	OUTPUT
SVM	INPUT (300)	281	11	0	7	1	0
	AND (94)	47	42	0	2	1	2
	OR (72)	65	5	0	2	1	2
	NOT (95)	75	2	0	14	4	0
	XOR (47)	39	3	0	4	1	0
	OUTPUT (70)	3	3	0	0	0	64
	NB	INPUT (300)	75	21	160	23	15
AND (94)		10	33	34	2	9	6
OR (72)		5	3	54	2	8	0
NOT (95)		7	6	41	19	21	1
XOR (47)		0	9	21	7	9	1
OUTPUT (70)		0	1	1	0	0	68
KNN		INPUT(300)	167	86	16	19	11
	AND (94)	29	47	6	3	6	3
	OR (72)	30	23	5	8	6	0
	NOT (95)	28	18	7	22	20	0
	XOR (47)	21	15	1	6	4	0
	OUTPUT (70)	1	5	0	0	0	64

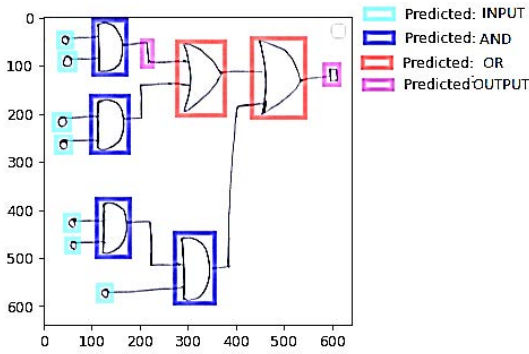
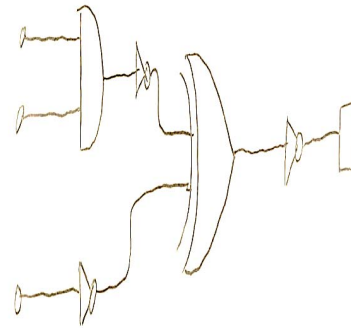
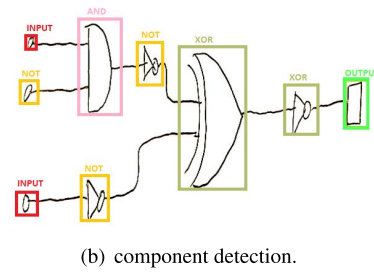


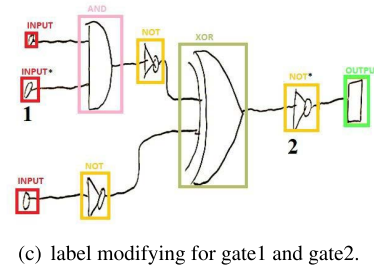
FIGURE 10. Error of gate detection in Faster R-CNN.



(a) original image.



(b) component detection.



(c) label modifying for gate1 and gate2.

xyz	$F=(x' \oplus (y.z))'$
000	1
001	1
010	1
011	0
100	0
101	0
110	0
111	1

(d) binary function and truth table.

FIGURE 12. Analyzing handwritten logic circuits.

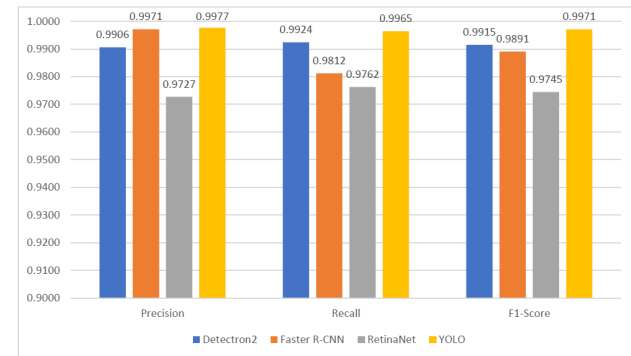


FIGURE 11. Precision, recall, and F1-score on the JSU-HWLC dataset.

STEP 7- TRUTH TABLE

In this step, to obtain the truth table we create a row for each input combination of the function. Then, for each gate we calculate its output value and then the function output. Table 2 shows the truth table made for the function of Figure 9. In this table, the first four columns represent the value of the inputs (x, y, w, z), and the last column represents the value of the function.

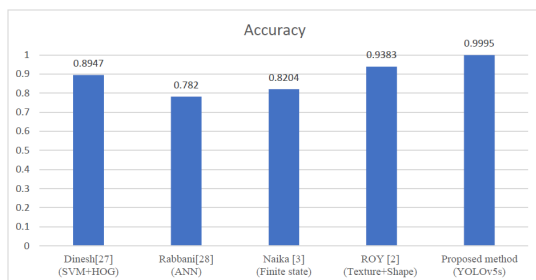
III. EXPERIMENTAL RESULTS

As mentioned in Section II, the object detection module of the proposed system was implemented using the YOLO version 5. YOLOv5 includes five different models: YOLOv5s (smallest), YOLOv5m, YOLOv5l, YOLOv5x (largest). The proposed system uses the YOLOv5s release. There was no need to use more complex and heavier models because YOLOv5s achieved high accuracy in the experiments. This algorithm was implemented using Google Colab services. Colab or Colaboratory is an interactive notebook provided by Google for writing and running Python through a browser.

In the training phase of YOLO on the Google Colab platform, no improvement was achieved after epoch 324. Table 3 shows the confusion matrix resulting from the YOLO and three other deep learning algorithms on the testing image set. The testing dataset includes 70 handwritten circuits with a total of 300 INPUT pins, 94 AND gates, 72 OR gates, 95 NOT gates, 47 XOR gates, and also 70 OUTPUT pins throughout the set. In these matrices, there is a column called Error, which represents gate detection that have occurred in

an unreal place. In other words, this column represents parts of the image (such as the background or connection signals) that have been mistakenly identified as a gate. An example of these errors generated by Faster R-CNN is shown in Figure 10. In this figure, a part of the connection signal is mistakenly identified as an output component. As it can be seen in Table 3, two other methods (Detectron2 and RetinaNet) have led to such errors, but in the experiments, YOLO did not generate such errors. For this reason, the YOLO algorithm is used as a component detector in the proposed system. Figure 11 shows the precision, recall, and F1-score of YOLO in comparison with the other deep learning methods. This





**FIGURE 13.** Comparison of the proposed method with some state-of-the-art methods.

**TABLE 5.** Average precision in gate detection on JSU-HWLC dataset.

Gates	Sharma [10] (R-CNN)	Proposed Method (YOLOV5s)
INPUT	0.9967	1.0000
AND	1.0000	1.0000
OR	1.0000	0.9863
NOT	0.9691	1.0000
XOR	0.9783	1.0000
OUTPUT	0.9859	1.0000
Average	0.9883	0.9977

figure illustrates the superiority of the YOLO algorithm in these criteria.

Table 4 shows the confusion metrics of traditional methods including support vector machine (SVM), Naïve Bayes (NB), and K-nearest neighbor (KNN). These three algorithms have employed the histogram of oriented gradient (HOG) descriptor as the feature vector. As it can be seen in this table, traditional methods are much less efficient than the deep learning algorithm (Table 3). Because logic gates are visually similar, they have not been accurately classified using conventional methods based on feature extraction. The use of complex feature vectors may increase the accuracy of these algorithms. But as demonstrated in Table 3, YOLO as a deep-learning approach that does not need a separate step for feature extraction is capable of classifying the components of logic circuits accurately. Although YOLO can detect various gates with high accuracy, in the proposed system the user can manually modify the incorrect detection so that the incorrect binary function will not be generated in the next stage. Figure 12 shows the different steps of the proposed system on a sample handwritten circuit. In this example circuit, two components were misdiagnosed by YOLO. The user modifies this error, and finally, the correct truth table is generated.

Comparison of the proposed hand-drawn circuit component recognition method with some state-of-the-art methods is shown in Figure 13. It should be noted that the accuracy of the proposed method has been calculated using the sample images of logic gates provided by ROY *et al.* [2], and the accuracy of the other methods is obtained from [2]. Table 5 shows the class-level and also average values of precision in comparison with an RCNN-based method [10]. To have a fair comparison, this method has been implemented on JSU-HWLC using the hyperparameters provided by Sharma [10].

#### IV. CONCLUSION

A new system for understanding handwritten logic circuits was provided in this article. In this method, the circuit components are first identified using a known algorithm in deep learning called YOLO. This algorithm is trained by a customized set of images (JSU-HWLC). Then, the connection among the detected components is investigated by a new simple boundary tracking method. Once the circuit components, including gates and the input and output pins and the connections between them were detected, the binary function of the handwritten logic circuit was obtained, and finally, the truth table was generated. The results of the experiments show the high efficiency of the proposed method. Therefore, this simple and efficient system can be used in sketching phase engineering projects as well as on smart training boards for teaching logical circuits and computer architecture.

The proposed system was tried to be as simple as possible and low cost in terms of processing load, so it can be easily performed on any ordinary platform. Since this system has a structural procedure, each module can be easily replaced with a more accurate method in the future. At present, the proposed system works correctly to process circuits that have simple wiring. It can detect all connection lines. So, the loss ratio, in this case, is zero. But it has some problems for complex circuits with multi-branch connections or intersecting lines. We will expand the dataset to include more complex circuits with crossing connections and multi-branch signals and also other digital components such as decoders, encoders, multiplexers, full adders, and flip-flops to improve the system. Our proposed system will be trained on the most commonly used letters in logic circuits such as x, y, z, w, a, b, c, d, etc. Adding reinforcement learning techniques is another future work of this research to modify detection errors, and achieve higher accuracy. Additional stages such as simplifying the circuit using the Karnaugh map and the automatic production of HDL code corresponding to the simplified function are considered as future goals of this system.

#### REFERENCES

- [1] I. Khan, N. Islam, H. Ur Rehman, and M. Khan, "A comparative study of graphic symbol recognition methods," *Multimedia Tools Appl.*, vol. 79, nos. 13–14, pp. 8695–8725, Apr. 2020, doi: [10.1007/s11042-018-6289-6](https://doi.org/10.1007/s11042-018-6289-6).
- [2] S. Roy, A. Bhattacharya, N. Sarkar, S. Malakar, and R. Sarkar, "Offline hand-drawn circuit component recognition using texture and shape-based features," *Multimedia Tools Appl.*, vol. 79, nos. 41–42, pp. 31353–31373, Nov. 2020, doi: [10.1007/s11042-020-09570-6](https://doi.org/10.1007/s11042-020-09570-6).
- [3] R. L. Naika, R. Dinesh, and S. Prabhanjan, "Handwritten electric circuit diagram recognition: An approach based on finite state machine," *Int. J. Mach. Learn. Comput.*, vol. 9, no. 3, pp. 374–380, Jun. 2019, doi: [10.18178/ijmlc.2019.9.3.813](https://doi.org/10.18178/ijmlc.2019.9.3.813).
- [4] M. Moetesum, S. Waqar Younus, M. Ali Warsi, and I. Siddiqi, "Segmentation and recognition of electronic components in hand-drawn circuit diagrams," *ICST Trans. Scalable Inf. Syst.*, vol. 5, no. 16, Apr. 2018, Art. no. 154478, doi: [10.4108/eai.13-4-2018.154478](https://doi.org/10.4108/eai.13-4-2018.154478).
- [5] X. Zhang, Y. Li, Z. Zhang, K. Konno, and S. Hu, "Intelligent Chinese calligraphy beautification from handwritten characters for robotic writing," *Vis. Comput.*, vol. 35, nos. 6–8, pp. 1193–1205, Jun. 2019, doi: [10.1007/s00371-019-01675-w](https://doi.org/10.1007/s00371-019-01675-w).

- [6] S. Ghosh, A. Chatterjee, P. K. Singh, S. Bhowmik, and R. Sarkar, "Language-invariant novel feature descriptors for handwritten numeral recognition," *Vis. Comput.*, vol. 37, no. 7, pp. 1781–1803, Jul. 2021, doi: [10.1007/s00371-020-01938-x](https://doi.org/10.1007/s00371-020-01938-x).
- [7] H. Guo, Y. Liu, D. Yang, and J. Zhao, "Offline handwritten Tai Le character recognition using ensemble deep learning," *Vis. Comput.*, Jul. 2021, doi: [10.1007/s00371-021-02230-2](https://doi.org/10.1007/s00371-021-02230-2).
- [8] J. Chaki and N. Dey, "Fragmented handwritten digit recognition using grading scheme and fuzzy rules," *Sādhanā*, vol. 45, no. 1, pp. 1–23, Dec. 2020, doi: [10.1007/s12046-020-01410-5](https://doi.org/10.1007/s12046-020-01410-5).
- [9] S. P. Deore and A. Pravin, "Devanagari handwritten character recognition using fine-tuned deep convolutional neural network on trivial dataset," *Sādhanā*, vol. 45, no. 1, pp. 1–13, Dec. 2020, doi: [10.1007/s12046-020-01484-1](https://doi.org/10.1007/s12046-020-01484-1).
- [10] M. Sharma, S. Nipane, Rachita, K. N. Jariwala, and R. Khade, "DLC re-builder: Sketch based recognition and 2-D conversion of digital logic circuit," in *Advanced Computing*, D. Garg, K. Wong, J. Sarangapani, and S. K. Gupta, Eds. Singapore: Springer, 2021, pp. 200–214, doi: [10.1007/978-981-16-0404-1\\_15](https://doi.org/10.1007/978-981-16-0404-1_15).
- [11] A. Abdallah, A. Berendeyev, I. Nuradin, and D. Nurseitov, "TNCR: Table net detection and classification dataset," *Neurocomputing*, vol. 473, pp. 79–97, Feb. 2022.
- [12] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017, doi: [10.1109/TPAMI.2016.2577031](https://doi.org/10.1109/TPAMI.2016.2577031).
- [13] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2980–2988.
- [14] W. Y. Yuxin, A. Kirillov, F. Massa, L. Wan-Yen, and R. Girshick. (2019). *Detectron2*. [Online]. Available: <https://github.com/facebookresearch/detectron2>
- [15] M. Dey, S. M. Mia, N. Sarkar, A. Bhattacharya, S. Roy, S. Malakar, and R. Sarkar, "A two-stage CNN-based hand-drawn electrical and electronic circuit component recognition system," *Neural Comput. Appl.*, vol. 33, no. 20, pp. 13367–13390, Oct. 2021.
- [16] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 779–788.
- [17] J. Tao, H. Wang, X. Zhang, X. Li, and H. Yang, "An object detection system based on Yolo in traffic scene," in *Proc. 6th Int. Conf. Comput. Sci. New Technol. (ICCSNT)*, Oct. 2017, pp. 315–319.
- [18] V. B. S. Mahalleh, T. A. AL Qutami, and I. A.-T. Mahmood, "YOLO-based valve type recognition and localization," in *Proc. IEEE 6th Int. Conf. Ind. Eng. Appl. (ICIEA)*, Apr. 2019, pp. 37–40.
- [19] C. Dewi, R.-C. Chen, Y.-T. Liu, X. Jiang, and K. D. Hartomo, "YOLO V4 for advanced traffic sign recognition with synthetic training data generated by various GAN," *IEEE Access*, vol. 9, pp. 97228–97242, 2021.
- [20] W. Chen, H. Huang, S. Peng, C. Zhou, and C. Zhang, "YOLO-face: A real-time face detector," *Vis. Comput.*, vol. 37, no. 4, pp. 805–813, Apr. 2021, doi: [10.1007/s00371-020-01831-7](https://doi.org/10.1007/s00371-020-01831-7).
- [21] M. H. Junos, A. S. Mohd Khairuddin, S. Thannirmalai, and M. Dahari, "Automatic detection of oil palm fruits from UAV images using an improved YOLO model," *Vis. Comput.*, vol. 38, pp. 2341–2355, Apr. 2021, doi: [10.1007/s00371-021-02116-3](https://doi.org/10.1007/s00371-021-02116-3).
- [22] S. Zhou, Y. Bi, X. Wei, J. Liu, Z. Ye, F. Li, and Y. Du, "Automated detection and classification of spilled loads on freeways based on improved Yolo network," *Mach. Vis. Appl.*, vol. 32, no. 2, pp. 1–12, Mar. 2021, doi: [10.1007/s00138-021-01171-z](https://doi.org/10.1007/s00138-021-01171-z).
- [23] Y. Li, Z. Han, H. Xu, L. Liu, X. Li, and K. Zhang, "YOLOv3-lite: A lightweight crack detection network for aircraft structure based on depthwise separable convolutions," *Appl. Sci.*, vol. 9, no. 18, p. 3781, Sep. 2019.
- [24] W. Min, X. Li, Q. Wang, Q. Zeng, and Y. Liao, "New approach to vehicle license plate location based on new model YOLO-L and plate pre-identification," *IET Image Process.*, vol. 13, no. 7, pp. 1041–1049, May 2019, doi: [10.1049/iet-ipr.2018.6449](https://doi.org/10.1049/iet-ipr.2018.6449).
- [25] Y. Tian, G. Yang, Z. Wang, H. Wang, E. Li, and Z. Liang, "Apple detection during different growth stages in orchards using the improved YOLO-V3 model," *Comput. Electron. Agricult.*, vol. 157, pp. 417–426, Feb. 2019.
- [26] S. Noor, M. Waqas, M. I. Saleem, and H. N. Minhas, "Automatic object tracking and segmentation using unsupervised SiamMask," *IEEE Access*, vol. 9, pp. 106550–106559, 2021, doi: [10.1109/ACCESS.2021.3101054](https://doi.org/10.1109/ACCESS.2021.3101054).
- [27] R. Dinesh, "Handwritten electronic components recognition: An approach based on HOG+ SVM," *J. Theor. Appl. Inf. Technol.*, vol. 96, no. 13, pp. 4020–4028, 2018.
- [28] M. Rabbani, R. Khoshkangini, H. S. Nagendraswamy, and M. Conti, "Hand drawn optical circuit recognition," *Proc. Comput. Sci.*, vol. 84, pp. 41–48, Jan. 2016, doi: [10.1016/j.procs.2016.04.064](https://doi.org/10.1016/j.procs.2016.04.064).



**SOMAIEH AMRAEE** received the B.S. degree in computer engineering from Alzahra University, Iran, in 2006, the M.S. degree in computer architecture from the Isfahan University of Technology, Iran, in 2010, and the Ph.D. degree from the University of Isfahan, Iran, in 2018. She is currently an Assistant Professor in electrical and computer engineering with the Jundi-Shapur University of Technology, Dezful, Iran. Her research interests include computer vision, machine learning, and digital hardware design.



**MARYAM CHINIPARDAZ** received the B.Sc. degree in computer engineering-software, the B.Sc. degree in information technology, and the M.Sc. and Ph.D. degrees in computer networking from the Amirkabir University of Technology (AUT), in 2009, 2010, 2012, and 2018, respectively. She is currently an Assistant Professor in electrical and computer engineering at the Jundi-Shapur University of Technology, Dezful, Iran. Her current research interests include optimization methods and machine learning techniques in multimedia networks.



**MOHAMMADALI CHAROOSAEI** received the B.Sc. degree in computer engineering from the Jundi-Shapur University of Technology, Dezful, Iran, in 2021. He has a solid and up-to-date understanding of machine learning and deep learning algorithm's theoretical concepts with implementation experience. His research interests include machine learning and deep learning, especially their applications in computer vision and image processing.



**MOHAMMAD AMIN MIRZAEI** is currently pursuing the bachelor's degree in computer engineering with the Jundi-Shapur University of Technology, Dezful, Iran. His research interests include computer vision, artificial intelligence, and android programming. He has three ACM membership.